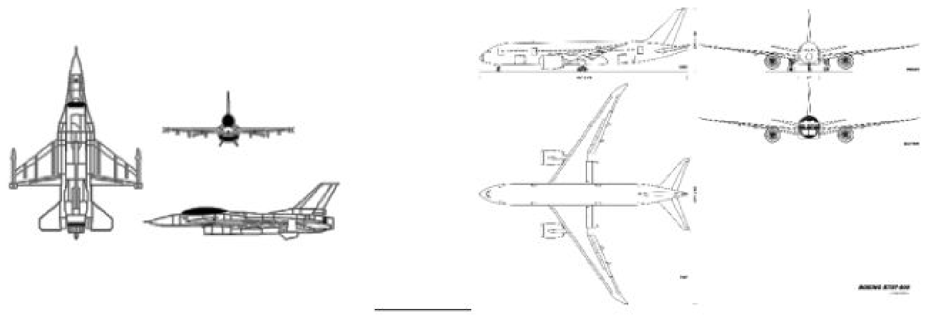
ASEN 2004 Introduction to Aerospace Vehicle Design and Performance Spring 2016

Experimental Laboratory 1: Low-Speed Aerodynamics of the Lockheed Martin F-16 and the Boeing 787 Dreamliner



ASEN 2004: Vehicle Design and Performance
Assigned: Wed 13 Jan 2016
Lab Reports Due: Thurs 3 March 2016, at the beginning of lecture

GROUP #29

Name: Justin Alvey
Wrote/Edited: Results, Discussion, Matlab Code: CL, CD, CM plots, error calculations, static margins, longitudinal stability, minimum landing speeds, L/D ratios, and error/herror plots

Name: Ankit Hriday
Wrote/Edited:

Name: Pierce Lieberman
Wrote/Edited: Introduction, Abstract, combined clean and dirty plots, Results, Discussion, Static Margins, Formatting.

Name: Brian Ortiz
Wrote/Edited: Introduction, Background and Theory, Discussion, Conclusion, and Formatting

ASEN 2004 Lab 1: Wind Tunnel Testing

Justin Alvey[1], Ankit Hriday[2], Pierce Lieberman[3], and Brian Ortiz[4]

ABSTRACT

In this lab we investigated the aerodynamic characteristics of the Lockheed Martin F-16 Fighting Falcon and the Boeing 787 Dreamliner in a low-speed environment. Two types of F-16's were tested: one was "clean" and the other was "dirty". The dirty one included external stores (e.g. instruments, weapons, and fuel tanks) and the clean one was just the plane. The aerodynamic characteristics of each of the F-16 configurations were compared against each other. This data was also compared with the Dreamliner data to find any overarching similarities. From this data, the static margin for each aircraft was calculated to determine stability of each model. The F-16 Fighting Falcon was found unstable, and the Dreamliner 787 found stable, and the additional of external stores proved to increase drag over the wing of the F-16 model.

Nomenclature

α	=	Angle of attack
A	=	Axial force acting on the sting
C_D	=	Coefficient of Drag
C_L	=	Coefficient of Lift
C_M	=	Coefficient of Moment
D	=	Drag
g	=	Gravitational acceleration
L	=	Lift
N	=	Normal force acting on the sting
ρ	=	Density at elevation
$P_{m_{cg}}$	=	Pitching moment about the model's center of gravity
P_m	=	Pitching moment about the sting balance center
S	=	Wing area
W	=	Total weight of the aircraft
σ	=	Standard deviation

[1] Student ID: 103265867
[2] Student ID: 104464022
[3] Student ID: 103089071
[4] Student ID: 102917432

I. Table of Contents

ASEN 2004 Lab 1

II. Introduction

In previous labs we analyzed the aerodynamic characteristics of airfoils which spanned the test section. In this lab the focus was more on analyzing the performance of an entire airplane not just an airfoil. For that reason two types of scaled aircraft were examined using a wind tunnel. The two aircraft included the Lockheed Martin F-16 Fighting Falcon, in its loaded and unloaded configurations, and the Boeing 787 Dreamliner. The two configurations of the F-16 included a "dirty" configuration and a "clean" one. The "dirty" configuration demonstrated a more realistic jet with instruments, weapons, etc. and the "clean" one did not. In order to perform design and performance analysis we calculated the lift coefficient, C_L, the drag coefficient, C_D, and the moment coefficient about the model's center of gravity. These were all calculated with respect to the angle of attack, α. With these coefficients calculated, the drag polars and longitudinal stability plots were generated, as well as C_L and C_D vs. α plots.

III. Background and Theory

The model planes were placed on a sting balance inside the wind tunnel. This sting balance calculates and separates normal, *N,* and axial, *A,* forces as well as the pitching moment that act on the model. These forces are used to calculate Lift, *L,* and Drag, *D,* components using the formulas in Equation 1.

$$\begin{aligned} D &= N\sin\alpha + A\cos\alpha \\ L &= N\cos\alpha - A\sin\alpha \end{aligned} \tag{1}$$

We were told to assume that the error at each sample set point was random so that sample values followed a normal distribution centered on the mean value $\overline{x}$ (Eq. 2). This allowed us to find a standard deviation for a small sample size of $n \le 20$ (Eq. 3).

$$\overline{x} = \frac{\sum_{i=1}^{n} x_i}{n} \tag{2}$$

$$\sigma = \left[\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2\right]^{1/2} \tag{3}$$

The pitching moment read by the LabVIEW VI is that of the pitching moment, P_m, about the sting's balance center. In order to find the pitching moment about the model's center of gravity (CG), $P_{m_{cg}}$, the moment created by the normal force with respect to the extra distance from the model's CG to the sting's balance center, $M(x)$, had to be subtracted from the VI data (Eq. 4). X is the distance from the model's CG to the balance center which is different for each model (Table 1).

$$P_{m_{cg}} = P_m - M(x) \tag{4}$$

Table 1 - Distance of Model's CG to Sting Balance Center

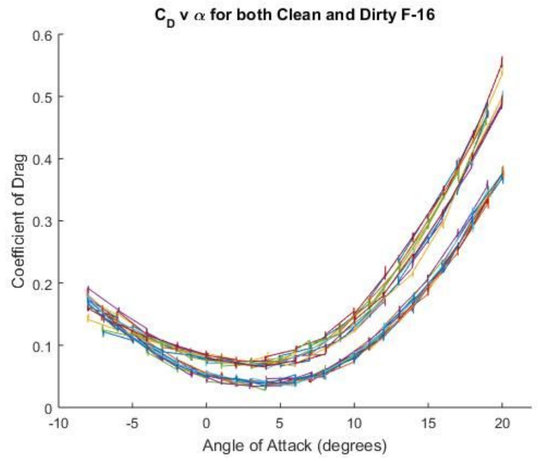
Model	***x (in mm)***
Clean F-16	14.4
Dirty F-16	15.5
Boeing 787	63.0

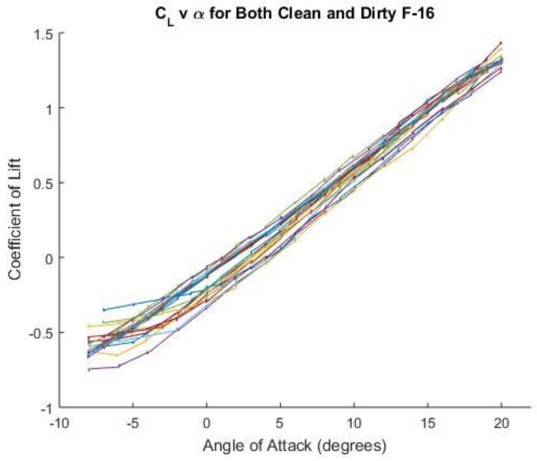
IV. Experimental Apparatus and Procedure

The experiment was conducted in a low-speed (< 65 m/s) wind tunnel with three models: a clean 1/45 scale model of an F-16, a "dirty" 1/45 scale model of an F-16 which included fuel tanks, missiles, and other external loads, along with a 1/255 scale model of a clean Boeing 787 Dreamliner. The models were mounted and leveled on a sting balance in the test section of the wind tunnel and set up to record the normal and axial forces and pitching moment of the model. Before testing began, the wind tunnel software was turned on and all sensors zeroed. All sensors were then turned on, recording the forces and pitching moment of the model as it was cycled up through angles of attack from -8 to 20 degrees at 0 wind speed, with some groups doing odd angles and the others doing evens. Once the data was recorded, the model was then lowered to the lowest testing angle of attack, and the wind tunnel switched on to a speed of 25 m/s. The angle of attack was then again raised incrementally up to the maximum angle of attack allowed, recording the forces and pitching moment at each angle. All data was then saved and the wind tunnel and LABView software shut down. Post-experiment, MATLAB was utilized to process the data for each model at each angle of attack to determine the aerodynamic characteristics (lift, drag, etc.) of each model, as well as the error associated with the measurements taken.

V. Summary of Results

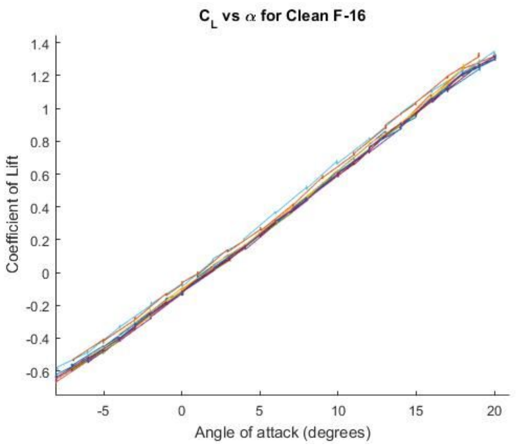
Testing of the clean and dirty F-16 models showed that the drag coefficient was hardly impacted by the addition of external stores at lower angles of attack. However, as the angle of attack increases, the effect of the external stores can be seen with an increasingly large difference between the drag coefficients of the clean and dirty models, with the dirty model having at the maximum almost a 0.15 increase from the clean model, as can be seen in Figure 1 below.

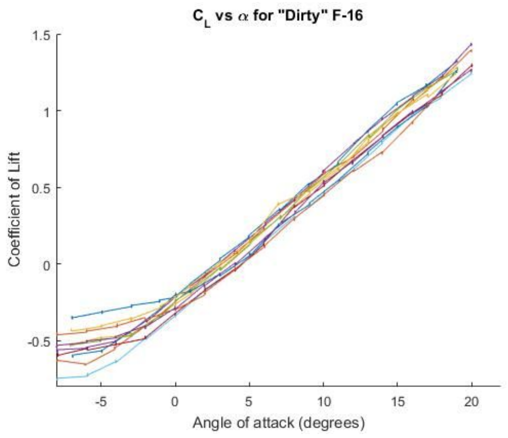




Figures 1, 2: The Lift and Drag Coefficients vs Angle of Attack for both clean and dirty F-16 models.

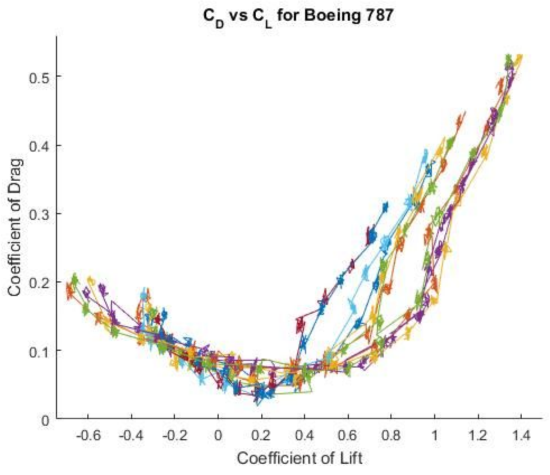
As can be seen in Figure 2 above, the lift coefficients for the clean and dirty models were relatively similar at higher angles of attack. With a decreasing angle of attack, it can be seen that the lift coefficient of the dirty model is slightly lower, and below an angle of attack of 0 degrees can be seen to be far more erratic between data sets, which is better illustrated in Figures 3 and 4 below, which show the lift coefficients for each model separately. Here it is much clearer that the data for the clean model remains much more linear and consistent across all angles of attack, whereas the dirty model is far more unpredictable at lower values.

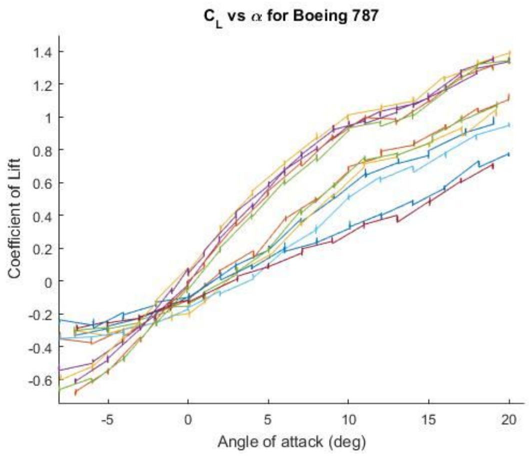




Figures 3, 4: Lift Coefficient vs Angle of Attack for clean and dirty F-16 models.

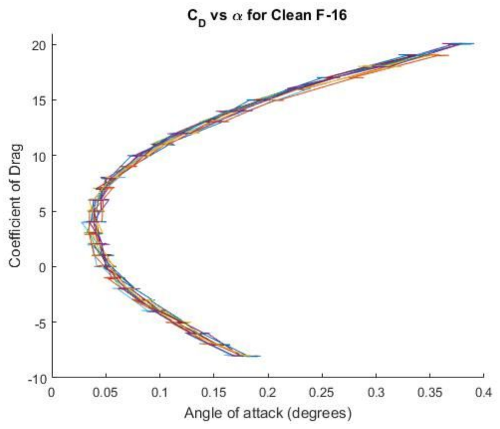
For Boeing 787 model, the *C_L* vs *C_D* and *C_L* vs plots are present below. Observing only the clean case of the Boeing 787, the data reflects consistent trends for all aerodynamic coefficients involved. The Boeing 787 creates about as much lift as the F-16, and slightly less drag than the other aircraft fighter jet. However, the wind-tunnel data retrieved for the Dreamliner is more chaotic in nature, at virtually all angles of attack. More accurate data should be taken for the Boeing 787 model to further explore the true nature of the aircraft in flight.

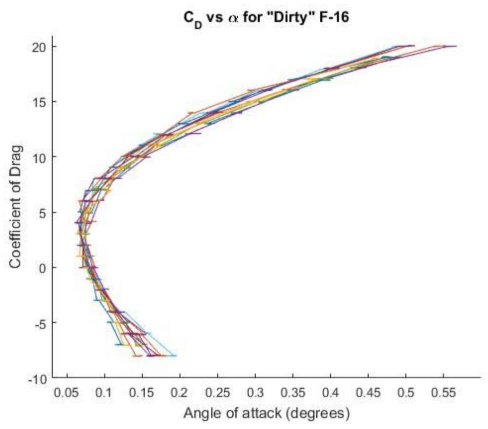




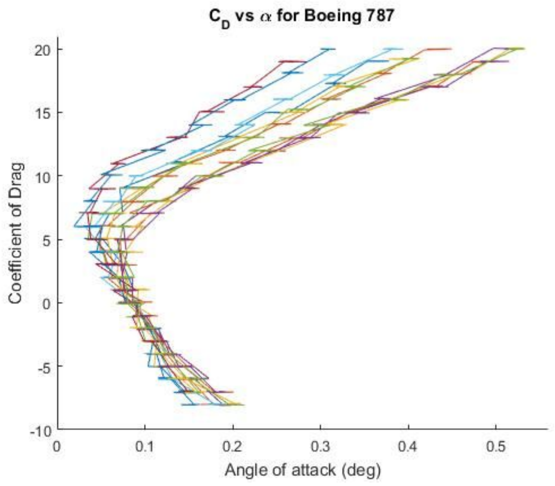
Figures 5, 6: The Lift and Drag Coefficients vs Angle of Attack for Boeing 787 model.

The drag polar (*C_D* vs *C_L*) plot is an important aerodynamic relationship that often requires a certain trade off: how the generation of lift affects the generation of drag for a given aircraft. A more subtle relationship is *C_D* vs ɑ, which shows a similar rendering, but at a lesser magnitude of scale. Shown below are Figures 7, 8, and 9, which are plots of *C_D* vs ɑ for the clean F-16, "dirty" F-16, and the Boeing 787. These plots show how the change in coefficient of drag with angle of attack.





Figures 7, 8, 9: Drag Coefficients vs Angle of Attack for both clean and dirty F-16 models and Boeing 787



The moment coefficient about the center of gravity showed a significant difference between the clean and dirty models, which is shown below in Figure 10. Here, while both data sets have a similar shape, with the moment coefficient increasing linearly before briefly levelling off at about a 15 degree angle of attack, they differ in slope and initial values.

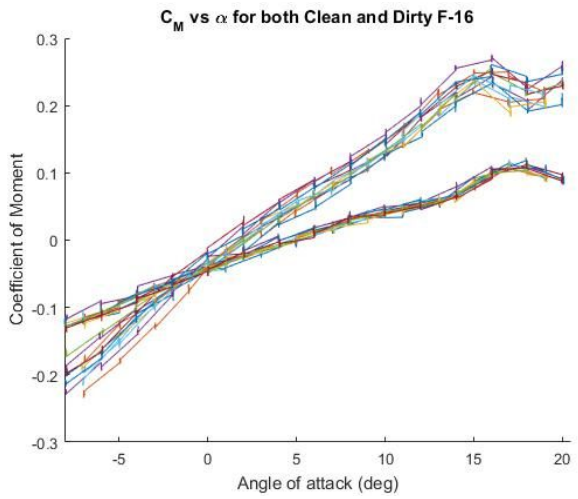
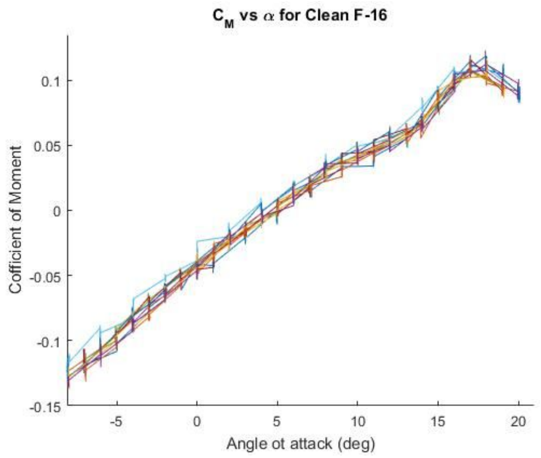
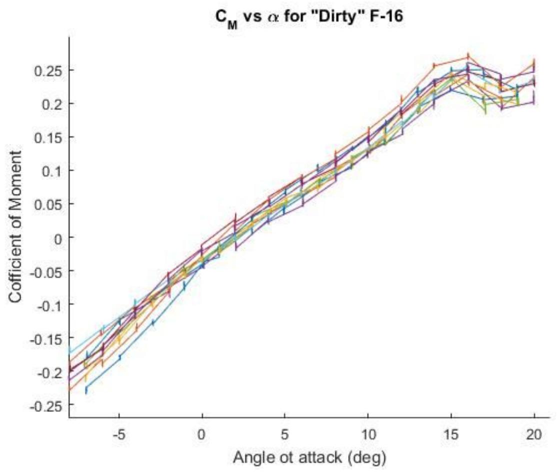


Figure 10: Moment Coefficients vs Angle of Attack for both clean and dirty F-16 models.

The clean model begins with a higher initial magnitude, but increases more slowly than that of the dirty model, before levelling off around a max value of 1.229. The dirty model ends at a much higher magnitude, around 0.2757. In the dirty model case, the moment coefficient even starts to increase yet again after hitting its max value. Notice the obvious uncertainty introduced by adding stores onto the F-16 model, given by the erratic aerodynamic behavior of the dirty model. Each relationship is below, Figure 11 and Figure 12, respectively.





Figures 11, 12: Moment Coefficients vs Angle of Attack for clean and dirty F-16 models.

A contrasting between the moment coefficient with angle of attack occurs for the Boeing 787 model. Seen in Figure 13, this model produces exhibits an overall negative slope for C_M, which makes sense for an aircraft that is designed for high stability. When the Boeing model is inclined at angle of attack, incoming air tries to displace the model to an even higher angle of attack. Due to the model's aerodynamic design, an increasingly negative moment is created about the model's center of gravity, which tends to bring the nose of the model down to equilibrium (angle of attack of 0º).

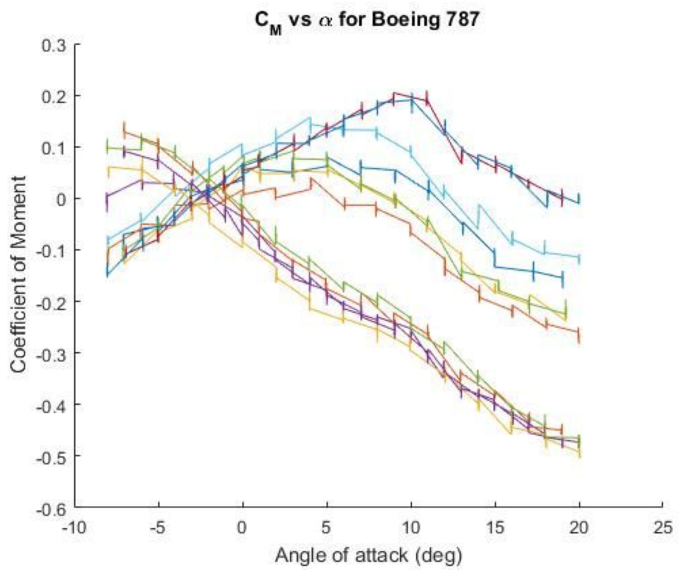


Figure 13: Moment Coefficients vs Angle of Attack for Boeing 787 model.

The aerodynamic quantity calculated next was the minimum landing speed of each model. For the F-16 Fighting Falcon, the minimum landing speed of each configuration was developed for the model, and also the minimum landing speed of the full-size aircraft. For the Dreamliner 787, the minimum landing speed of the model was calculated at the conditions of the lab, and the minimum landing speed of the full-size aircraft. The landing speeds of the models and the actual designs are presented below (Table 2).

Table 2 - Landing Speeds of Model and Real Aircraft

Model Design	***Minimum Landing Speed Calculated (m /s)***
F-16 Model: Clean	6.095
F-16 Model: Dirty	5.900
Boeing 787 Dreamliner: Clean	6.719
F-16 Aircraft: Full-Size	98.710
Boeing 787 Dreamliner: Full-Size	106.604

The minimum landing speed of the F-16 was found by first calculating Vstall. The stall speed of an aircraft is defined as the speed at which an aircraft can no longer fly, without risking the lives on board the aircraft. Beneath this speed, the aircraft cannot sustain flight. The effective stall speed of an aircraft can be calculated as below in Eq (5):

$$V stall = \sqrt{2\ W/(\rho\ S\ CL)} \tag{5}$$

where ρ is the density of the air at the altitude in which the aircraft is flying. The minimum landing speed of an F-16 is used in industry as 1.2 Vstall. The minimum landing speed for a Boeing 787 is used in industry as 1.3 Vstall.

At an angle of attack of zero degrees, it was found that the slope of the moment coefficients (longitudinal stability) for the clean F-16 model was found to be 0.0085/degree, and the slope of the moment coefficient plot was found to be 0.0723/degree. Using Eq (6) below:

$$\text{Static Margin} = \frac{-\frac{\delta C_M}{\delta \alpha}}{\frac{\delta C_L}{\delta \alpha}} \tag{6}$$

it was calculated that the static margin of the clean F-16 model at an angle of attack of zero was -11.76%. It was also found that the longitudinal stability of the 787 model was -0.011/degree, and the slope of the lift coefficients at zero angle of attack was 0.0625/degree, which using Eq (6) resulted in a static margin of 17.6%. All of these values are tabulated in Table 3 for convenience.

Table 3 - Stability Results for Each Model

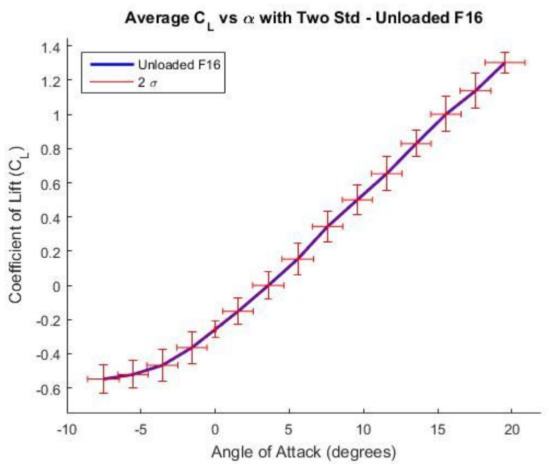
Model	$\delta CM/\ \delta\alpha$	$\delta CL/\ \delta\alpha$	***Static Margin***
F-16 Clean	0.0085	0.0723	-0.1176
F-16 Dirty	0.0166	0.0731	-0.2273
787 Dreamliner	-0.0110	0.0625	0.1760

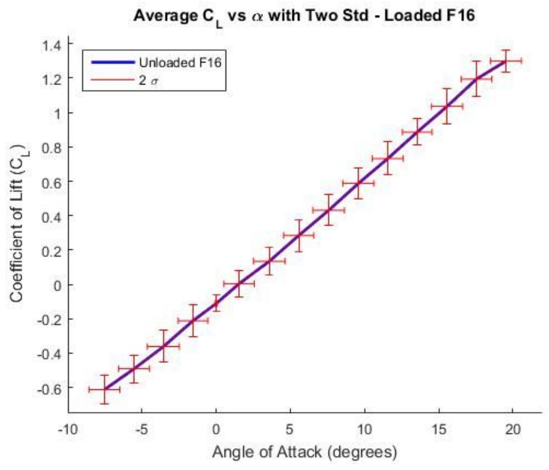
The lift-to-drag ratio of an aircraft is an aerodynamic quantity used to determine the efficiency of an aircraft. With a high lift-to-drag ratio, an aircraft can conserve more fuel, as it conversely takes less fuel to create lift on ascent, and requires less fuel to overcome drag associated with flight conditions. Eq (7) is the equation used to calculate the maximum lift-to-drag ratio for an aircraft in flight.

$$L/D\ max = 1/2 \sqrt{\pi\, e\, AR/\ CD,o} \tag{7}$$

This ratio is a function of the span efficiency factor, aspect ratio, and the zero-lift drag coefficient. For both the F-16 clean model and the Boeing 787 model, the lift-to-drag ratio was calculated based on the wind tunnel data. The F-16 model had a lift-to-drag ratio of 3.473, and the Boeing 787 had a lift-to-drag ratio of 2.637.

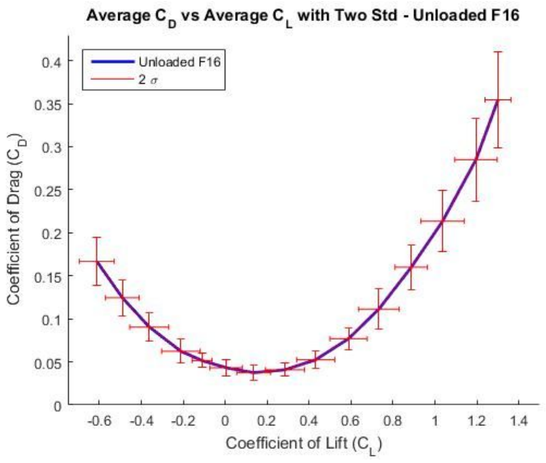
Error analysis was performed on the resulting wind tunnel data to determine its validity. For each relationship, C_D vs C_L, C_D vs α, etc., the standard deviation at each data point was calculated based on random error, using Eq (2) and Eq (3). Error bars were then calculated for each plot, given by two standard deviations. In Figures 14 and 15 below are calculated error bars, against the mean, for C_L vs α for both the clean and “dirty” F-16 configurations. The error bars and mean distribution are fairly similar, which is expected, since the plots varied so little, seen in Figures 2, 3, and 4.

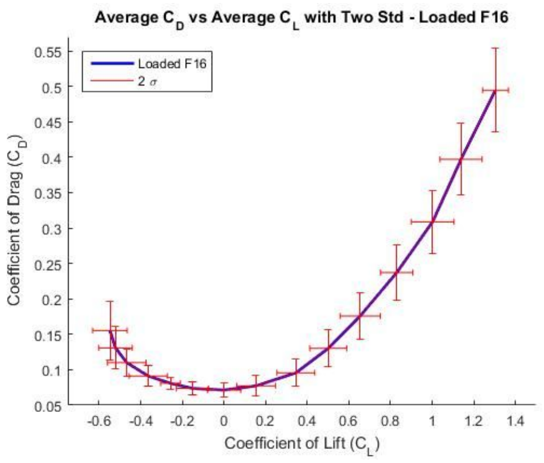




Figures 14, 15: Mean Lift Coefficients vs Angle of Attack with Error Bars, for Clean and Dirty F-16 Models

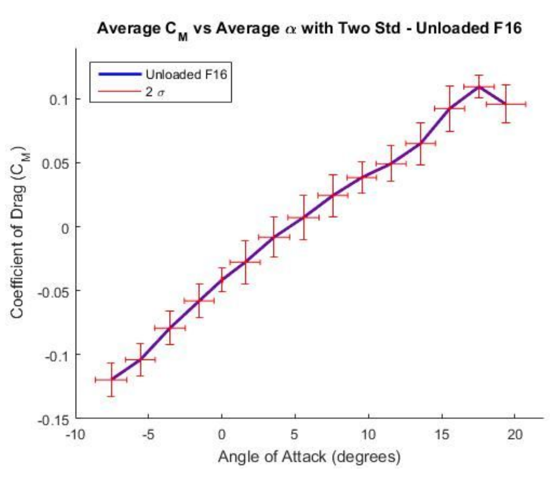
The same error calculations were performed for the drag polars in each case, shown below in Figures 16,17. These plots have more pronounced values for their error bars. More specifically, the drag is more affected by random error than lift at high angles of attack, and negative angles of attack. The error associated with drag calculated at low angles of attack (-2 to 2) is fairly small.

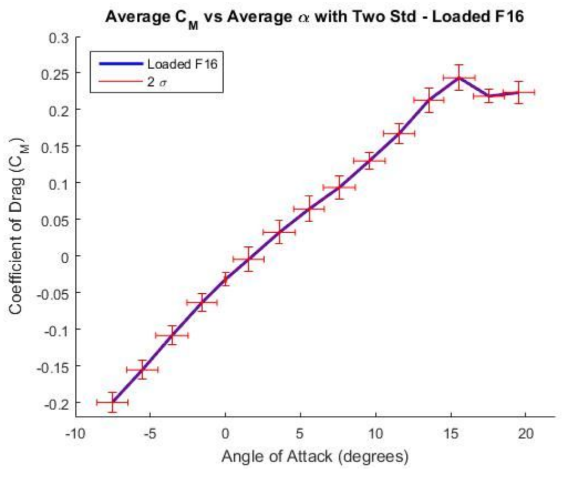




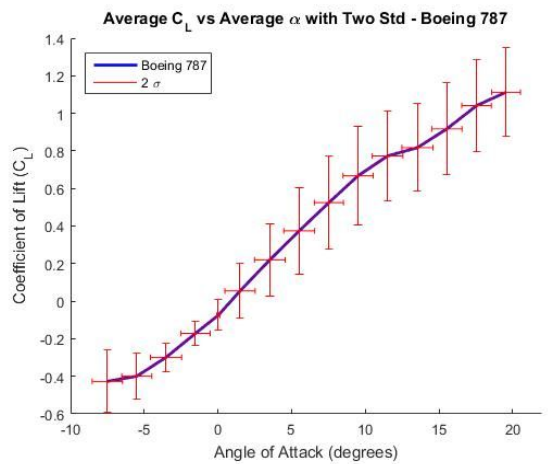
Figures 16, 17: Mean Drag Coefficients vs Angle of Attack with Error Bars, for Clean and Dirty F-16 Models

The following error calculations were performed several more times, for the *CM* vs plots for both F-16 models (18, 19), and the remaining plots for the Boeing 787 (20, 21, 22). In Figures 18 and 19, notice the low error associated with values at low angles of attack (-1 to 1).





Figures 18, 19: Mean Moment Coefficients vs Angle of Attack with Error Bars, for Clean and Dirty F-16 Models



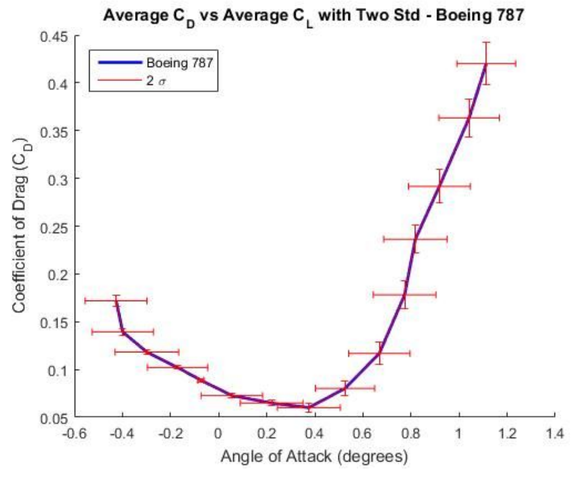
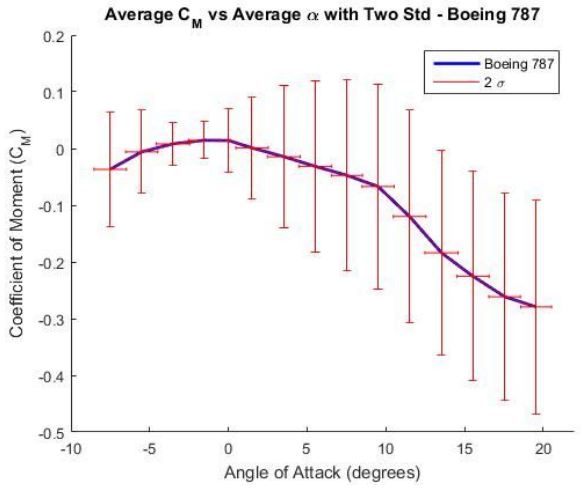


Figure 20, 21, 22: Mean Lift Coefficients, Drag Coefficients, and Moment Coefficients vs Angle of Attack with Error Bars, for Boeing 787 model



The data retrieved for the Boeing 787 is relevant, in that relationships between the aerodynamic quantities during testing can be demonstrated. However, based on the error calculations involved and the high level of uncertainty in Figures 20, 21, 22 above, the authors have stressed caution in trusting this data, beyond educational purposes.

VI. Discussion

Adding external stores increases the drag coefficient from that of the clean model F-16. This occurs as the missiles, fuel tanks, etc are not as aerodynamic as a pure wing which leads to an increase in drag on the aircraft. The difference between the clean and dirty models is less pronounced at lower and negative angles of attack as the shape of the wing partially shields the external stores from the airflow, while at higher angles of attack these stores are directly in the flow, causing turbulence. It is the turbulence and flow disruptions caused by these stores that results in a slightly lower lift coefficient for the dirty model compared to the clean. At lower angles of attack, they can cause significant flow separation, resulting in the erratic behaviour of the data at low angles of attack in Figure 4. It is based on this behavior that stores are not carried on upper surface of a wing. The difference in moment coefficient between the two configurations is due to the exposed external stores further decreasing the stability of the dirty model as it increases its angle of attack, whereas the clean model is less adversely affected.

The minimum landing speed for each configuration of the F-16 Fighting Falcon was calculated (Table 2). For the "Dirty" F-16 model case, the minimum landing speed was much lower than the "Clean" case. Mathematically, this is due to the fact that the maximum coefficient of lift for the "Dirty" case was larger than the "Clean" case. By the addition of external stores, the overall drag on the model increased at varying angles of attack, along with a slight increase in the coefficient of lift overall. This makes sense because with an increase of the coefficient of lift (maximum), the stall velocity will decrease. The minimum landing speed for the Boeing 787 model, full-size F-16, and full-size Boeing 787 can also be found in Table 2. The Boeing requires a higher minimum landing speed, 208 knots (107 m/s) for a safe landing versus 192 knots (99 m/s) for the F-16. Since the Boeing weighs much more than the F-16 and has a larger wingspan, it requires a higher landing speed on landing. For this lab, an assumption was made for a "clean" landing of the aircraft, which did not account for the effects of flaps, gear drag, and thrust reversal on landing.

The clean F-16 model had a far lower static margin than that of the Boeing 787 model. For both of the F-16 configurations, the calculated static margin was negative, while the static margin was positive for the Boeing 787. This relationship is as expected, as a large civilian transport is going to have an airframe based heavily around stability and lifting power. A fighter jet such as the F-16 is designed to be far less stable, so as to facilitate quick, high-g maneuvers. This enables a pilot to pursue or evade another aircraft in combat, and not have to fight with an aircraft that would prefer to return to equilibrium after even a slight displacement. Neither case is necessarily right or wrong, as aircraft design is based on mission specifics. Since the static margin serves as a factor by which to determine static longitudinal stability, the F-16 can be considered an longitudinally unstable aircraft, while the Boeing 787 is considered stable, although both accomplish their desired goals. In other terms, the Boeing 787 exhibits more longitudinal static stability than the F-16.

Based on the wind tunnel data, the F-16 clean model appeared to exhibit a better efficiency at sea-level flight conditions than the Boeing 787. Although the Boeing 787 has a much larger aspect ratio, the F-16 model still performed better in creating lift versus drag. From this, a conclusion was made involving other characteristics of the aircraft's design. The Boeing 787 created more lift than the F-16, but also generated more drag due to its size and enormous fuselage. So even though the Boeing 787 had a higher aspect ratio, its zero-lift drag coefficient was much higher. Based on the ratio in Eq (7), the F-16 exhibits a higher lift-to-drag ratio, because it introduced less drag overall, versus slightly more lift generated by the Boeing 787.

VII. Conclusion

Using the wind tunnel, we were able to analyze models of the Lockheed Martin F-16 Fighting Falcon and the Boeing 787 Dreamliner. Two models were used for the F-16, a "clean" one and "dirty" one. The results of the wind tunnel experiment showed that the "dirty" F-16 model with all its external instruments and stores saw a higher coefficient of drag than the "clean" model. These results are shown and backed by Figure 1. The static margin analysis showed that the F-16 models were unstable, but the 787 Dreamliner model was stable. The data gather in this experiment could be optimized and perfected by using a different sting balance with the least amount of error as possible. The error in the results is propagated when terms are added or subtracted. A case where this happened was when the pitching moment of the model was changed due to the difference in the distance between the model's CG and the balance center. This could be avoided if the sting balance center and the model's CG are the same. Overall the results showed what was expected between the aircraft models.

VIII. References

[1] Dr. Gerren, D. ‘ASEN 2004 Experiment 1 SP16_Feb2’ D2L, Univ. of Colorado-Boulder, Boulder, CO, 2016 (unpublished)

[2]. Introduction to Flight. John D Anderson, 8th Edition

[3] Modern Combat Aircraft Design, Klaus Huenecke, 1st Edition

IX. Appendix

ExpLab1_Group29.m - Main File

```
%% Original Author: Justin Alvey
% Lab: Experimental Lab 2004 - F-16 and 787 Windtunnel Exp.
% Date Created: 1/14/16
% Date Last Edited: 3/1/15

clc;
clear all;
close all;

%% Test Data
test1 = 'F16_CLEAN_LA.csv';
test2 = 'F16_CLEAN_LA2.csv';

%% Constants for Calculation
scaleF16 = 1/48; % Scale of the F16
scale787 = 1/225; % Scale of the 787
rho_Boulder = 1.0425; % Density at Boulder (kg/m^3)
pound2N = 4.44822162; % Conversion from lb to N

%% F16 Wing Area
wing_area_actual = 27.87; % Reference:
http://www.lockheedmartin.com/us/products/f16/F-16Specifications.ht
ml
wing_area = wing_area_actual*(scaleF16^2); % Wing area sized-down
WeightF16 = 133446.64800000002; % Landing weight of the full-scale
F16 (N)

%% F16 Chord
% Cited From "Modern Combat Aircraft Design" by Klaus Huenecke:
F_16_root_chord = 5.04; % Actual length
F_16_tip_chord =  1.0583; % Actual length
F_16_chord = scaleF16*(F_16_root_chord + F_16_tip_chord)/2; %
Scaled-down chord (F16)

%% Boeing 787 Chord
% Cited From: http://www.lissys.demon.co.uk/samp1/
root_787_chord = 38.94; % Actual Length
```

```
tip_787_chord = 5.55; % Actual Length
BoeingChord = scale787*(root_787_chord + tip_787_chord)/2; %
Scaled-down chord (787)

%% 787 Wing Area
% Cited From:
https://en.wikipedia.org/wiki/General_Dynamics_F-16_Fighting_Falcon
wing_area_787_actual = 325; % Area of the 787 wing
wing_area_787 = wing_area_787_actual*(scale787^2); % Scaled-down
wing area of the 787
Weight787 = 1601359.776; % Landing weight of the Boeing 787 (N)

%% Load F16 Clean Data Files from Directory (Current Folder)
[CLclean, CDclean, CMclean,
wind_aoa_clean,k,wind_rho,weightmodelF16,error_CLclean,
error_CDclean] = Cleanload(wing_area,F_16_chord); % Calls
Cleanload.m

%% Load F16 Loaded Data Files from Directory (Current Folder)
[CLdirty, CDdirty, CMdirty, wind_aoa_dirty,c,error_CLdirty,
error_CDdirty] = Dirtyload(wing_area,F_16_chord); % Calls
Dirtyload.m

%% Load Boeing 787 Clean Data Files from Directory (Current Folder)
[CL787clean, CD787clean, CM787clean,
wind_aoa_787clean,d,wind_rho_787,weightmodel787] =
BoeingClean(wing_area_787,BoeingChord);
% Calls BoeingClean.m

%% Error Calculations
%Only use -5 to 5 degrees because past that the data is not as
linear, so
%it will throw off the best fit line a little
%aoa = linspace(-5,5,11); % Angle of attack for polyfit
calculations
%alpha = linspace(-8,20,29); % Angle of attack for average graphs

alpha_clean_std = std(wind_aoa_clean);
alpha_dirty_std = std(wind_aoa_dirty);

CL_clean_std = std(CLclean);
CD_clean_std = std(CDclean);
CL_dirty_std = std(CLclean);
CD_dirty_std = std(CDdirty);

% Normalizing the Data for Each Standard Deviation
for i=0:14
```

```
    alpha_clean_stds(i*20+1:20+i*20) =
mean(alpha_clean_std(i*20+1:20+i*20));
    alpha_dirty_stds(i*20+1:20+i*20) =
mean(alpha_dirty_std(i*20+1:20+i*20));
    CL_clean_stds(i*20+1:20+i*20) =
mean(CL_clean_std(i*20+1:20+i*20));
    CLclstd(i+1) = CL_clean_stds(i*20+1);
    CL_dirty_stds(i*20+1:20+i*20) =
mean(CL_dirty_std(i*20+1:20+i*20));
    CD_clean_stds(i*20+1:20+i*20) =
mean(CD_clean_std(i*20+1:20+i*20));
    CD_dirty_stds(i*20+1:20+i*20) =
mean(CD_dirty_std(i*20+1:20+i*20));
end

for i= [2:20 22:40 42:60 62:80 82:100 102:120 122:140 142:160
162:180 182:200 202:222 224:240 242:260 262:280 282:300]
    for j=1:15
    if CL_clean_stds(i) == CLclstd(j)
        alpha_clean_stds(i) = 0;
        alpha_dirty_stds(i) = 0;
        CL_clean_stds(i) = 0;
        CL_dirty_stds(i) = 0;
        CD_clean_stds(i) = 0;
        CD_dirty_stds(i) = 0;

    end
    end
end

% Locate the array elements in the std array that are not repeated
% Find returns the index of the values greater than 0, which are
the singly
% repeated values

alphacleanfind = find(alpha_clean_stds>0);
alphadirtyfind = find(alpha_dirty_stds>0);
CLcleanfind = find(CL_clean_stds>0);
CLdirtyfind =  find(CL_dirty_stds>0);
CDcleanfind =  find(CD_clean_stds>0);
CDdirtyfind =  find(CD_dirty_stds>0);

% Reassign the non-repeated std values to a new array
finalalphacleanstd = alpha_clean_stds(alphacleanfind);
finalalphadirtystd = alpha_dirty_stds(alphadirtyfind);
finalCLcleanstd = CL_clean_stds(CLcleanfind);
finalCLdirtystd = CL_dirty_stds(CLdirtyfind);
```

```
finalCDcleanstd = CD_clean_stds(CDcleanfind);
finalCDdirtystd = CD_dirty_stds(CDdirtyfind);


%% Aerodynamic Calculations for the Clean F16
fprintf('1. Stability Data for the Clean F16\n');

% Calculating Static Longitudinal Stability - Clean F16
% Assumption Made: the The relationship for CL vs alpha is "nearly"
linear
% for the angles of attack used
coeff_clean = polyfit(wind_aoa_clean, CMclean,1); % Coefficients
for the polyfit of the CM data
dCM = coeff_clean(1); % dCm/d_alpha when alpha = 0 is the first
coefficient of the polyfit coefficients
fprintf('dCm/d(alpha) for Clean F16: %0.4f \n',dCM) % Format the
values for the user

% Calculating dCL/d(alpha) for Clean F16
% Assumption Made: the The relationship for CM vs alpha is "nearly"
linear
% for the angles of attack used
coeff_clean = polyfit(wind_aoa_clean,CLclean,1);
dCL = coeff_clean(1);
fprintf('dCL/d(alpha) for Clean F16: %0.4f \n',dCL) % Format the
values for the user

% Calculating the Static Margin (-dCM/d(alpha) / dCL/d(alpha)) -
Clean F16
staticMargin = -dCM/dCL;
fprintf('-(dCM/d(alpha))/(dCL/d(alpha)) for Clean F16: %0.4f
\n',staticMargin) % Format the values for the user
fprintf('\n')

% Calculating the Minimum Landing Speed - Clean F16
Vstall =
sqrt((2*max(weightmodelF16))./(mean(mean(wind_rho))*wing_area*max(m
ax(CLclean))));
fprintf('2. Minimum Landing Speed of the Clean F16 Model: Vmin =
%0.3f m/s \n',1.2*Vstall)
fprintf('\n')

%% Aerodynamic Calculations for the Dirty F16
fprintf('3. Stability Data for the Dirty F16\n');

% Calculating Static Longitudinal Stability - Dirty F16
coeff_clean = polyfit(wind_aoa_dirty, CMdirty,1); % Coefficients
```

```
for the polyfit of the CM data
dCM = coeff_clean(1); % dCm/d_alpha when alpha = 0 is the first
coefficient of the polyfit coefficients
fprintf('dCm/d(alpha) for Dirty F16: %0.4f \n',dCM) % Format the
values for the user

% Calculating dCL/d(alpha) - Dirty F16
coeff_clean = polyfit(wind_aoa_dirty,CLdirty,1);
dCL = coeff_clean(1);
fprintf('dCL/d(alpha) for Dirty F16: %0.4f \n',dCL) % Format the
values for the user

% Calculating the Static Margin (-dCM/d(alpha) / dCL/d(alpha)) -
Dirty F16
staticMargin = -dCM/dCL;
fprintf('-(dCM/d(alpha))/(dCL/d(alpha)) for Dirty F16: %0.4f
\n',staticMargin) % Format the values for the user
fprintf('\n')

% Calculating the Minimum Landing Speed - Dirty F16
Vstall =
sqrt((2*max(weightmodelF16))./(mean(mean(wind_rho))*wing_area*max(m
ax(CLdirty))));
fprintf('4. Minimum Landing Speed of the Dirty F16 Model: Vmin =
%0.3f m/s \n',1.2*Vstall)
fprintf('\n')

% Calculating the Minimum Landing Speed of the Full-Scale F16
Vstall_fullsize =
sqrt((2*WeightF16)./(rho_Boulder*wing_area_actual*max(max(CLclean))
));
fprintf('5. Minimum Landing Speed of the Full-size F16: Vmin =
%0.3f m/s \n',1.2*Vstall_fullsize)
fprintf('\n')

%% Aerodynamic Calculations for the Boeing 787
fprintf('6. Stability Data for the Boeing 787 \n');

% Calculating Static Longitudinal Stability - Boeing 787
coeff_dirty = polyfit(wind_aoa_787clean, CM787clean,1); %
Coefficients for the polyfit of the CM data
dCM = coeff_dirty(1); % dCm/d_alpha when alpha = 0 is the first
coefficient of the polyfit coefficients
fprintf('dCm/d(alpha) for Boeing 787: %0.4f \n',dCM) % Format the
values for the user

% Calculating dCL/d(alpha) - Boeing 787
```

```
coeff_clean = polyfit(wind_aoa_787clean, CL787clean,1);
dCL = coeff_clean(1);
fprintf('dCL/d(alpha) for Boeing 787: %0.4f \n',dCL) % Format the
values for the user

% Calculating the Static Margin (-dCM/d(alpha) / dCL/d(alpha)) -
Boeing 787
staticMargin = -dCM/dCL;
fprintf('-(dCM/d(alpha))/(dCL/d(alpha)) for Boeing 787: %0.4f
\n',staticMargin) % Format the values for the user
fprintf('\n')

% Calculating the Minimum Landing Speed of the Boeing 787
Vstall =
sqrt((2*max(weightmodel787))./(mean(mean(wind_rho_787))*wing_area_7
87*max(max(CL787clean))));
fprintf('7. Minimum Landing Speed of the Model Boeing 787: Vmin =
%0.3f m/s \n',1.3*Vstall)
fprintf('\n')

% Calculating the Minimum Landing Speed of the Actual Boeing 787
Vstall =
sqrt((2*Weight787)./(rho_Boulder*wing_area_787_actual*max(max(CL787
clean))));
fprintf('8. Minimum Landing Speed of the Actual Boeing 787: Vmin =
%0.3f m/s \n',1.3*Vstall)
fprintf('\n')

% Calculate the Lift/Drag ratio of the F-16
CLF16avg = max(max(CLclean));
CDF16avg = max(max(CDclean));
lift_drag = CLF16avg/CDF16avg;% Calculate the lift-to-drag ratio
fprintf('9. Lift/Drag Ratio of the F-16: L/D = %0.3f \n',lift_drag)
fprintf('\n')

% Calculate the Lift/Drag ratio of the Boeing 787
CL787avg = max(max(CL787clean));
CD787avg = max(max(CD787clean));
lift_drag = CL787avg/CD787avg;% Calculate the lift-to-drag ratio
fprintf('10. Lift/Drag Ratio of the Boeing 787: L/D = %0.3f
\n',lift_drag)
fprintf('\n')

%% Plotting - Clean F16
% Plot CL versus alpha before error analysis
figure(1);
   for j = 1:k
```

```
        hold on
        plot(wind_aoa_clean(j,:),CLclean(j,:));
        xlabel('Angle of attack (degrees)')
        ylabel('Coefficient of Lift')
        title('C_{L} vs \alpha for Clean F-16')
        xlim([min(min(wind_aoa_clean)) 21])
        ylim([-0.75 1.45])
    end

% Plot CD versus alpha before error analysis
figure(2);
    for j = 1:k
        hold on;
        plot(CDclean(j,:),wind_aoa_clean(j,:));
        xlabel('Angle of attack (degrees)')
        ylabel('Coefficient of Drag')
        title('C_{D} vs \alpha for Clean F-16')
        ylim([-10 21])
    end

    % Plot CD versus CL before error analysis
    figure(8);
    for j = 1:k
        hold on;
        plot(CDclean(j,:),CLclean(j,:));
        xlabel('Coefficient of Drag')
        ylabel('Cofficient of Lift')
        title('C_{D} vs C_{L} for Clean F-16')
        xlim([0 0.4])
        ylim([-0.75 1.45])
    end

    figure(3);
    % Plot CM versus alpha before error analysis
    for j = 1:k
        hold on;
        plot(wind_aoa_clean(j,:),CMclean(j,:));
        xlabel('Angle ot attack (deg)')
        ylabel('Cofficient of Moment')
        title('C_{M} vs \alpha for Clean F-16')
        xlim([min(min(wind_aoa_clean)) 21])
        ylim([-0.15 0.135])
    end

%% Plotting - Dirty F16
% Plot CL versus alpha before error analysis
figure(4);
```

```
   for j = 1:c
       hold on
       plot(wind_aoa_dirty(j,:),CLdirty(j,:));
       xlabel('Angle of attack (degrees)')
       ylabel('Coefficient of Lift')
       title('C_{L} vs \alpha for "Dirty" F-16')
       xlim([min(min(wind_aoa_dirty)) 21])
       ylim([-0.8 1.5])
   end

figure(5);
% Plot CD versus alpha before error analysis
   for j = 1:c
       hold on;
       plot(CDdirty(j,:),wind_aoa_dirty(j,:));
       xlabel('Angle of attack (degrees)')
       ylabel('Coefficient of Drag')
       title('C_{D} vs \alpha for "Dirty" F-16')
       ylim([-10 21])
       xlim([0.03 0.6])
   end

   figure(6);
   % Plot CD versus CL before error analysis
   for j = 1:c
       hold on;
       plot(CDdirty(j,:),CLdirty(j,:));
       xlabel('Coefficient of Lift')
       ylabel('Cofficient of Drag')
       title('C_{D} vs C_{L} for "Dirty" F-16')
       xlim([0.03 0.6])
       ylim([-0.85 1.5])
   end

   figure(7);
   % Plot CM versus alpha before error analysis
   for j = 1:c
       hold on;
       plot(wind_aoa_dirty(j,:),CMdirty(j,:));
       xlabel('Angle ot attack (deg)')
       ylabel('Cofficient of Moment')
       title('C_{M} vs \alpha for "Dirty" F-16')
       xlim([min(min(wind_aoa_dirty)) 21])
       ylim([-0.27 0.3])
   end

%% Plotting - Boeing 787
```

```
% Plot CL versus alpha before error analysis
figure(9);
for i=1:d
    hold on;
    plot(wind_aoa_787clean(i,:),CL787clean(i,:))
    xlabel('Angle of attack (deg)')
    ylabel('Coefficient of Lift')
    title('C_{L} vs \alpha for Boeing 787')
    xlim([min(min(wind_aoa_787clean)) 21])
    ylim([-0.75 1.5])

end

% Plot CD versus alpha before error analysis
figure(10);
for i=1:d
    hold on;
    plot(CD787clean(i,:),wind_aoa_787clean(i,:))
    xlabel('Angle of attack (deg)')
    ylabel('Coefficient of Drag')
    title('C_{D} vs \alpha for Boeing 787')
    ylim([-10,21])
    xlim([0 0.56])

end


% Plot CD versus CL before error analysis
figure(11);
for i=1:d
    hold on;
    plot(CL787clean(i,:),CD787clean(i,:))
    xlabel('Coefficient of Lift')
    ylabel('Coefficient of Drag')
    title('C_{D} vs C_{L} for Boeing 787')
    xlim([min(min(CL787clean))-0.05,1.5])
    ylim([0 0.56])

end

% Plot CM versus alpha before error analysis
figure(12);
for i=1:d
    hold on;
    plot(wind_aoa_787clean(i,:),CM787clean(i,:))
    xlabel('Angle of attack (deg)')
    ylabel('Coefficient of Moment')
    title('C_{M} vs \alpha for Boeing 787')
    %xlim([min(min(wind_aoa_787clean)),21])
    %ylim([-10 22])
end

figure(13);
for i=1:c
    hold on;
    plot(wind_aoa_clean(i,:),CMclean(i,:))
    plot(wind_aoa_dirty(i,:),CMdirty(i,:))
    xlabel('Angle of attack (deg)')
    ylabel('Coefficient of Moment')
    title('C_{M} vs \alpha for both Clean and Dirty F-16')
    xlim([min(min(wind_aoa_clean)),20.5])
end

%% Error Calculations
m = mean(CLclean);
n = mean(CLdirty);
o = mean(wind_aoa_clean);
p = mean(wind_aoa_dirty);

% Average CL vs alpha for Clean and Dirty Cases (2 stds)
figure
for i=1:c
hold on
xlabel('Angle of Attack (degrees)')
ylabel('Coefficient of Lift (C_L)')
title(['Average C_{L} vs \alpha with One' ...
    ' Std - Unloaded F16'])
plot(p(CLdirtyfind),n(CLdirtyfind),'b','LineWidth',2)
errorbar(p(CLdirtyfind),n(CLdirtyfind), 2*finalCLcleanstd,'r')
herrorbar(p(CLdirtyfind),n(CLdirtyfind), 2*finalalphacleanstd,'r')
legend('Unloaded F16', '2 \sigma', 'Location', 'Northwest')
ylim([-1 1.45])
end

% Average CL vs alpha for Dirty Case (2 stds)
figure
for i=1:c
hold on
xlabel('Angle of Attack (degrees)')
ylabel('Coefficient of Lift (C_L)')
title(['Average C_{L} vs \alpha with One' ...
    ' Std - Loaded F16'])
plot(p(CLdirtyfind),m(CLdirtyfind),'b','LineWidth',2)
errorbar(p(CLdirtyfind),m(CLdirtyfind), 2*finalCLdirtystd,'r')
```

```
herrorbar(p(CLdirtyfind),m(CLdirtyfind), 2*finalalphadirtystd,'r')
legend('Unloaded F16', '2 \sigma', 'Location', 'Northwest')
ylim([-1 1.45])
end

u = mean(CDclean);
v = mean(CDdirty);

% Average CD vs CL for Clean Case (2 stds)
figure
for i=1:c
hold on
xlabel('Coefficient of Lift (C_{L})')
ylabel('Coefficient of Drag (C_{D})')
title(['Average C_{D} vs Average C_{L} with One' ...
    ' Std - Unloaded F16'])
plot(m(CDcleanfind),u(CDcleanfind),'b','LineWidth',2)
%plot(n(CDdirtyfind),v(CDdirtyfind),'b','LineWidth',2)
errorbar(m(CDcleanfind),u(CDcleanfind), 2*finalCDcleanstd,'r')
%errorbar(n(CDdirtyfind),v(CDdirtyfind), 2*finalCDdirtystd,'b')
herrorbar(m(CDcleanfind),u(CDcleanfind), 2*finalCLcleanstd,'r')
%herrorbar(n(CDdirtyfind),v(CDdirtyfind), 2*finalCLdirtystd,'b')
legend('Unloaded F16', '2 \sigma', 'Location', 'Northwest')
end

% Average CD vs CL for Dirty Case (2 stds)
figure
for i=1:c
hold on
xlabel('Coefficient of Lift (C_{L})')
ylabel('Coefficient of Drag (C_{D})')
title(['Average C_{D} vs Average C_{L} with One' ...
    ' Std - Loaded F16'])
plot(n(CDdirtyfind),v(CDdirtyfind),'b','LineWidth',2)
errorbar(n(CDdirtyfind),v(CDdirtyfind), 2*finalCDdirtystd,'r')
herrorbar(n(CDdirtyfind),v(CDdirtyfind), 2*finalCLdirtystd,'r')
legend('Loaded F16', '2 \sigma', 'Location', 'Northwest')
end

%% END OF CODE
```

Cleanload.m

```
function
```

```
[CL,CD,CM,wind_aoa,k,wind_rho,no_wind_N_force,std_q,error_CL,error_
CD] = Cleanload(wing_area,F_16_chord)
%% All "Clean" Cases
k = 0;
for i = [1 2 3 4 13 14 15 16 25 26 28]
    k = k + 1;
if(i >= 10)
    f_16 = strcat('F16_Clean_G',num2str(i),'.csv');
else
    f_16 = strcat('F16_Clean_G0',num2str(i),'.csv');
end

%% Load Data File
file = load(f_16);

%% Store data from all no wind cases
no_wind_pressure = file(1:300,1); % Find the atmosph. pressure
no_wind_A_force = file(1:300,25); % Find the axial force acting on
the sting
no_wind_N_force = file(1:300,24); % Find normal force acting on the
sting
no_wind_aoa = file(1:300,23); % Find angle of attack from file
no_wind_pitch_moment = file(1:300,26); % Find pitching moment
acting on the sting
no_wind_v_inf = file(1:300,4);
no_wind_rho = file(1:300,3);
distance = 0.0144; % Distance from the sting in m

%% Store data from all wind cases
wind_pressure(k,:) = file(301:600,1); % Find the atmosph. pressure
wind_A_force = file(301:600,25); % Find the axial force acting on
the sting
wind_N_force = file(301:600,24); % Find the normal force acting on
the sting
wind_aoa(k,:) = file(301:600,23); % Find the angle of attack
wind_pitch_moment = file(301:600,26); % Find the pitching moment
acting on the sting
wind_v_inf(k,:) = file(301:600,4); % Wind speed
wind_rho(k,:) = file(301:600,3); % Wind air density
wind_q_inf(k,:) = file(301:600,5); % Wind Dynamic Pressure

%% Calculate Aerodynamic Properties from Data
Normal(k,:) = wind_N_force - no_wind_N_force; % Normal Force
mat = Normal(k,:)*distance;
Axial(k,:) = wind_A_force - no_wind_A_force; % Axial Force
Moment(k,:) = wind_pitch_moment- no_wind_pitch_moment -
transpose(mat); % Pitching Moment
```

```
Lift(k,:) =
Normal(k,:).*cosd(wind_aoa(k,:))-Axial(k,:).*sind(wind_aoa(k,:)); %
Lift Force
Drag(k,:) =
Normal(k,:).*sind(wind_aoa(k,:))+Axial(k,:).*cosd(wind_aoa(k,:)); %
Drag Force
%Alpha = linspace(-8, 20, 299);

%% Calculate CL, CD, and CM
CL(k, :) = Lift(k,:) ./ (wind_q_inf(k, :) * wing_area);
CD(k, :) = Drag(k,:) ./ (wind_q_inf(k, :) * wing_area);
CM(k,:) = Moment(k,:) ./ (wind_q_inf(k, :) * wing_area *
F_16_chord);

% Calculate Standard Deviation of data for comparison
std_alpha_clean = std(wind_aoa(k,:));
std_normal_clean = std(Normal(k,:));
std_axial_clean = std(Axial(k,:));
std_moment_clean = std(Moment(k,:));
std_v_inf = mean(std(wind_v_inf(k,:)));

std_q(k, :) = sqrt((wind_rho(k,:).*wind_v_inf(k,
:)).^2.*std_v_inf.^2);

std_lift(k, :) = sqrt(cosd(wind_aoa(k, :)).^2.*std_normal_clean.^2
+ ...
        (sind(wind_aoa(k, :)).^2).*std_axial_clean.^2 + ...
        (Normal(k, :).*sind(wind_aoa(k, :)) - ...
        Axial(k, :).*cosd(wind_aoa(k, :))).^2.*std_alpha_clean);

    std_drag(k, :) = sqrt(sind(wind_aoa(k,
:)).^2.*std_normal_clean.^2 + ...
        (cosd(wind_aoa(k, :)).^2).*std_axial_clean.^2 + ...
        (Normal(k, :).*cosd(wind_aoa(k, :)) + ...
        Axial(k, :).*sind(wind_aoa(k, :))).^2.*std_alpha_clean);

    error_CL(k, :) = sqrt(((1./(wind_q_inf(k, :).* ...
        wing_area)).^2).*std_lift(k, :).^2 + ...
        (-Lift(k,:)./(wind_q_inf(k, :).*wing_area).^2).^2.*std_q(k,
:).^2);

    error_CD(k, :) = sqrt(((1./(wind_q_inf(k, :).* ...
        wing_area)).^2).*std_drag(k, :).^2 + ...
        (-Drag(k,:)./(wind_q_inf(k, :).*wing_area).^2).^2.*std_q(k,
:).^2);
end
```

```
end
```

Dirtyload.m

```
function [CL,CD,CM,wind_aoa,k,error_CL,error_CD] =
Dirtyload(wing_area,F_16_chord)
%% All "Dirty" Cases
k = 0;
for i = [5 6 7 8 17 18 20 29 30 31 32]
    k = k + 1;
if(i >= 10)
    f_16 = strcat('F16_Loaded_G',num2str(i),'.csv');
else
    f_16 = strcat('F16_Loaded_G0',num2str(i),'.csv');
end

%% Load Data Appropriate File
file = load(f_16);


%% Store data from all no wind cases
no_wind_pressure = file(1:300,1); % Find the atmosph. pressure
no_wind_A_force = file(1:300,25); % Find the axial force acting on
the sting
no_wind_N_force = file(1:300,24); % Find normal force acting on the
sting
no_wind_aoa = file(1:300,23); % Find angle of attack from file
no_wind_pitch_moment = file(1:300,26); % Find pitching moment
acting on the sting
no_wind_v_inf = file(1:300,4);
no_wind_rho = file(1:300,5);

% Store data from all wind cases
wind_pressure(k,:) = file(301:600,1); % Find the atmosph. pressure
wind_A_force = file(301:600,25); % Find the axial force acting on
the sting
wind_N_force = file(301:600,24); % Find the normal force acting on
the sting
wind_aoa(k,:) = file(301:600,23); % Find the angle of attack
wind_pitch_moment = file(301:600,26); % Find the pitching moment
acting on the sting
wind_rho = file(301:600,3);
wind_v_inf(k,:) = file(301:600,4);
wind_q_inf(k,:) = file(301:600,5);
distance = 0.0155; % Distance from the sting in m
```

```
%% Calculate Aerodynamic Properties from Data
Normal(k,:) = wind_N_force - no_wind_N_force; % Normal Force
mat = Normal(k,:)*distance;
Axial(k,:) = wind_A_force - no_wind_A_force; % Axial Force
Moment(k,:) = wind_pitch_moment- no_wind_pitch_moment-
transpose(mat); % Pitching Moment

Lift(k,:) =
Normal(k,:).*cosd(wind_aoa(k,:))-Axial(k,:).*sind(wind_aoa(k,:)); %
Lift Force
Drag(k,:) =
Normal(k,:).*sind(wind_aoa(k,:))+Axial(k,:).*cosd(wind_aoa(k,:)); %
Drag Force
%Alpha = linspace(-8, 20, 299);

%% Calculate CL, CD, and CM
CL(k, :) = Lift(k,:) ./ (wind_q_inf(k, :) * wing_area);
CD(k, :) = Drag(k,:) ./ (wind_q_inf(k, :) * wing_area);
CM(k,:) = Moment(k,:) ./ (wind_q_inf(k, :) * wing_area *
F_16_chord);

% Calculate Standard Deviation of data for comparison
std_alpha_clean = std(wind_aoa(k,:));
std_normal_clean = std(Normal(k,:));
std_axial_clean = std(Axial(k,:));
std_moment_clean = std(Moment(k,:));
std_v_inf = mean(std(wind_v_inf(k,:)));

std_q(k, :) = sqrt((wind_rho(k,:).*wind_v_inf(k,
:)).^2.*std_v_inf.^2);

std_lift(k, :) = sqrt(cosd(wind_aoa(k, :)).^2.*std_normal_clean.^2
+ ...
        (sind(wind_aoa(k, :)).^2).*std_axial_clean.^2 + ...
        (Normal(k, :).*sind(wind_aoa(k, :)) - ...
        Axial(k, :).*cosd(wind_aoa(k, :))).^2.*std_alpha_clean);

    std_drag(k, :) = sqrt(sind(wind_aoa(k,
:)).^2.*std_normal_clean.^2 + ...
        (cosd(wind_aoa(k, :)).^2).*std_axial_clean.^2 + ...
        (Normal(k, :).*cosd(wind_aoa(k, :)) + ...
        Axial(k, :).*sind(wind_aoa(k, :))).^2.*std_alpha_clean);

    error_CL(k, :) = sqrt(((1./(wind_q_inf(k, :).* ...
        wing_area)).^2).*std_lift(k, :).^2 + ...
        (-Lift(k,:)./(wind_q_inf(k, :).*wing_area).^2).^2.*std_q(k,
:).^2);

    error_CD(k, :) = sqrt(((1./(wind_q_inf(k, :).* ...
        wing_area)).^2).*std_drag(k, :).^2 + ...
        (-Drag(k,:)./(wind_q_inf(k, :).*wing_area).^2).^2.*std_q(k,
:).^2);
end


end
```

BoeingClean.m

```
function [CL,CD,CM,wind_aoa,k, wind_rho,no_wind_N_force] =
BoeingClean(wing_area,BoeingChord)
k = 0;
for i = [9 10 11 12 21 22 23 24 33 34 35 36]
    k = k + 1;
% All Boeing Cases (Only studying the clean case)
if(i >= 10)
    Boeing = strcat('787_G',num2str(i),'.csv');
else
    Boeing = strcat('787_G0',num2str(i),'.csv');
end

% Load File
file = load(Boeing);

% Store data from all no wind cases
no_wind_pressure = file(1:300,1); % Find the atmosph. pressure
no_wind_A_force = file(1:300,25); % Find the axial force acting on
the sting
no_wind_N_force = file(1:300,24); % Find normal force acting on the
sting
no_wind_aoa = file(1:300,23); % Find angle of attack from file
no_wind_pitch_moment = file(1:300,26); % Find pitching moment
acting on the sting
no_wind_v_inf = file(1:300,4);
no_wind_rho = file(1:300,3);
distance = 0.063; % Distance from the sting in m

% Store data from all wind cases
wind_pressure(k,:) = file(301:600,1); % Find the atmosph. pressure
wind_A_force = file(301:600,25); % Find the axial force acting on
the sting
wind_N_force = file(301:600,24); % Find the normal force acting on
```

```
the sting
wind_aoa(k,:) = file(301:600,23); % Find the angle of attack
wind_pitch_moment = file(301:600,26); % Find the pitching moment
acting on the sting
wind_v_inf(k,:) = file(301:600,4);
wind_rho = file(301:600,3);
wind_q_inf(k,:) = file(301:600,5);

% Calculate Aerodynamic Properties from Data
Normal(k,:) = wind_N_force - no_wind_N_force; % Normal Force
mat = Normal(k,:)*distance; % Moment about the sting to subtract
Axial(k,:) = wind_A_force - no_wind_A_force; % Axial Force
Moment(k,:) = wind_pitch_moment - no_wind_pitch_moment -
transpose(mat); % Pitching Moment about COG

Lift(k,:) =
Normal(k,:).*cosd(wind_aoa(k,:))-Axial(k,:).*sind(wind_aoa(k,:)); %
Lift Force
Drag(k,:) =
Normal(k,:).*sind(wind_aoa(k,:))+Axial(k,:).*cosd(wind_aoa(k,:)); %
Drag Force
%Alpha = linspace(-8, 20, 299);

% Calculate CL, CD, and CM
CL(k, :) = Lift(k,:) ./ (wind_q_inf(k, :) * wing_area);
CD(k, :) = Drag(k,:) ./ (wind_q_inf(k, :) * wing_area);
CM(k,:) = Moment(k,:) ./ (wind_q_inf(k, :) * wing_area *
BoeingChord);
end
end
```