% House Cleaning

clc

close all

clear vars

% Read in Port Locations

x = readmatrix('ClarkY14\_PortLocations.xlsx');

x(10,:)= []; % Remove NaN row

spanwise\_loc = [1;2;3;4;5;6;7;8;9;9.4375;9.875]; % Spanwise port locations

spanwise\_meters = spanwise\_loc \* 0.0254; % Convert to meters

y\_loc\_inf = x(:,3); % Port Locations -- y-axis

z\_loc\_inf = x(:,4); % z-axis

% Add in

y\_loc\_inf1 = [y\_loc\_inf(end);y\_loc\_inf(1:9);1;y\_loc\_inf(10:end)];

z\_loc\_inf1 = [z\_loc\_inf(end);z\_loc\_inf(1:9);0;z\_loc\_inf(10:end)];

chord\_l = 3.5031; % Chord length of wing

chord\_l\_m = 0.0254; % Chord length converted to meters

yc = y\_loc\_inf/chord\_l; % Normalize the chord length

% c\_p Function Call

[c\_p\_f,q\_f] =calculate\_c\_p(yc,1,'FullSpanTest.csv');

% Preallocate

C\_L\_f = zeros(1,length(spanwise\_meters));

C\_D\_f = zeros(1,length(spanwise\_meters));

w\_0 = zeros(1,length(spanwise\_meters));

% For Loop to Calculate C\_l and C\_d for Every Spanwise Position

for i = 1:length(spanwise\_loc)

[C\_L\_f(i),C\_D\_f(i),w\_0(i)] = calculateC\_l\_d(q\_f,y\_loc\_inf1,z\_loc\_inf1,yc,chord\_l\_m,chord\_l,spanwise\_loc(i),'FullSpanTest.csv');

end

% Plot Data

% C\_L vs Distance

figure(1);

subplot(1,2,1)

plot(spanwise\_loc,C\_L\_f,'.-b')

xlabel('Spanwise Position [in]')

grid on

ylabel('Coefficient of Lift')

title('Spanwise C\_L Along Wing')

% w(x) vs Distance

subplot(1,2,2)

plot(spanwise\_meters,w\_0,'.-b')

xlabel('Spanwise Position [m]')

ylabel('Spanwise Lift Distribution [N/m]')

title('Spanwise Lift Distribution')

grid on

function [c\_p,q\_ave] = calculate\_c\_p(l,spanwise,filename)

f\_data = load(filename);

logical\_spanwise = f\_data(:,9) == spanwise; % Uses data corresponding to aoa

f\_data\_spanwise = f\_data(logical\_spanwise,:);

rho\_f = f\_data\_spanwise(:,3); % Free-stream density

V\_f = f\_data\_spanwise(:,4); % Free-stream airspeed

q\_f = 0.5 .\* rho\_f .\* V\_f .^ 2;

q\_ave = mean(q\_f); % Average to get 1 value for dynamic pressure

delta\_P\_f = f\_data\_spanwise(:,15:30);

% Interpolate for Trailing Edge

delta\_P89 = delta\_P\_f(:,8) + (1-l(8))/(l(9)-l(8)) \* (delta\_P\_f(:,9)-delta\_P\_f(:,8));

delta\_P1011 = delta\_P\_f(:,10) + (1-l(10))/(l(11)-l(10)) \* (delta\_P\_f(:,11)-delta\_P\_f(:,10));

delta\_P\_ave = (delta\_P89 + delta\_P1011)/2; % Average both pressure readings

% Calculate c\_p values

c\_p = delta\_P\_f ./ q\_f;

c\_p\_TE = delta\_P\_ave ./ q\_f; % Trailing Edge

c\_p = [c\_p(:,1:9),c\_p\_TE,c\_p(:,10:end)]; % Add in interpolated data

c\_p = mean(c\_p); % Average the values

c\_p = [c\_p(end),c\_p]; % Close the loop

end

function [c\_l,c\_d,w\_0] = calculateC\_l\_d(q\_ave,y,z,l,chord\_m,chord,spanwise,filename)

% c\_p Function Call

c\_p1 = calculate\_c\_p(l,spanwise,filename);

% Integration Using trapz()

c\_n = -1/chord \* trapz(y,c\_p1); % Normal

c\_a = 1/chord \* trapz(z,c\_p1); % Axial

% Compute c\_l and c\_d

c\_l = c\_n \* cosd(10) - c\_a \* sind(10);

c\_d = c\_n \* sind(10) + c\_a \* cosd(10);

% Compute w\_0

w\_0 = c\_l \* q\_ave \* chord\_m;

end