Iowa State University

AERODYNAMIC FORCE MEASUREMENT ON AN ICING AIRFOIL

AER E 344 - Lab 12 - Aerodynamic Force Measurement on an Icing Airfoil

Section 3 Group 3

Matthew Mehrtens

Jack Mendoza

Kyle Ostendorf

Gabriel Pederson

Lucas Tavares Vasconcellos

Drew Taylor

Professor

Hui Hu, PhD

College of Engineering

Aerospace Engineering

Aerodynamics and Propulsion Laboratory

Ames, April 2024

Abstract

The presence of ice on wings can change the aerodynamic properties of a wing, so it is important that we understand the effects. By measuring the aerodynamic forces before and after the accretion of ice, we can quantify and predict the effect of icing on performance of aerodynamic bodies. Using force transducers, we are able to collect measurements to quantify aerodynamic forces. Comparing the aerodynamic forces between a clean airfoil and an iced airfoil, we were able to predict how a plane or a wing might respond if an aircraft underwent icing.

Contents

List of Figures

List of Tables

Glossary

α	angle of attack. *(p. 9, 10)*
$\frac{c_l}{c_d}$	lift to drag ratio. *(p. 3, 6, 9, 10)*
c_d	dimensionless coefficient of drag. *(p. 3, 9, 10)*
c_l	dimensionless coefficient of lift. *(p. 3, 9, 10)*
c_m	dimensionless coefficient of moment. *(p. 3, 6, 9, 10)*

Acronyms

AoA	angle of attack. *(p. 3, 7)*
MATLAB	MATrix LABoratory. *(p. 3)*

1

Introduction

In this lab, we looked at understanding how environmental factors affect aerodynamics and how it is crucial to understand these effects. One significant factor is ice forming on the wings of an aircraft. Ice formation can alter the wing aerodynamics, potentially causing a loss in lift and increased drag. To explore this, we used force and torque transducers in the ISU-UTAS Icing Research Tunnel to analyze aerodynamic forces on an airfoil before the ice was added to the wing and after the ice formed on the wings. The experiment aims to quantify these forces and show the impact of icing on aerodynamic performance. This research enhances our comprehension of aviation challenges caused by icing conditions (Hu, 2024).

2 Methodology

2.1 Apparatus

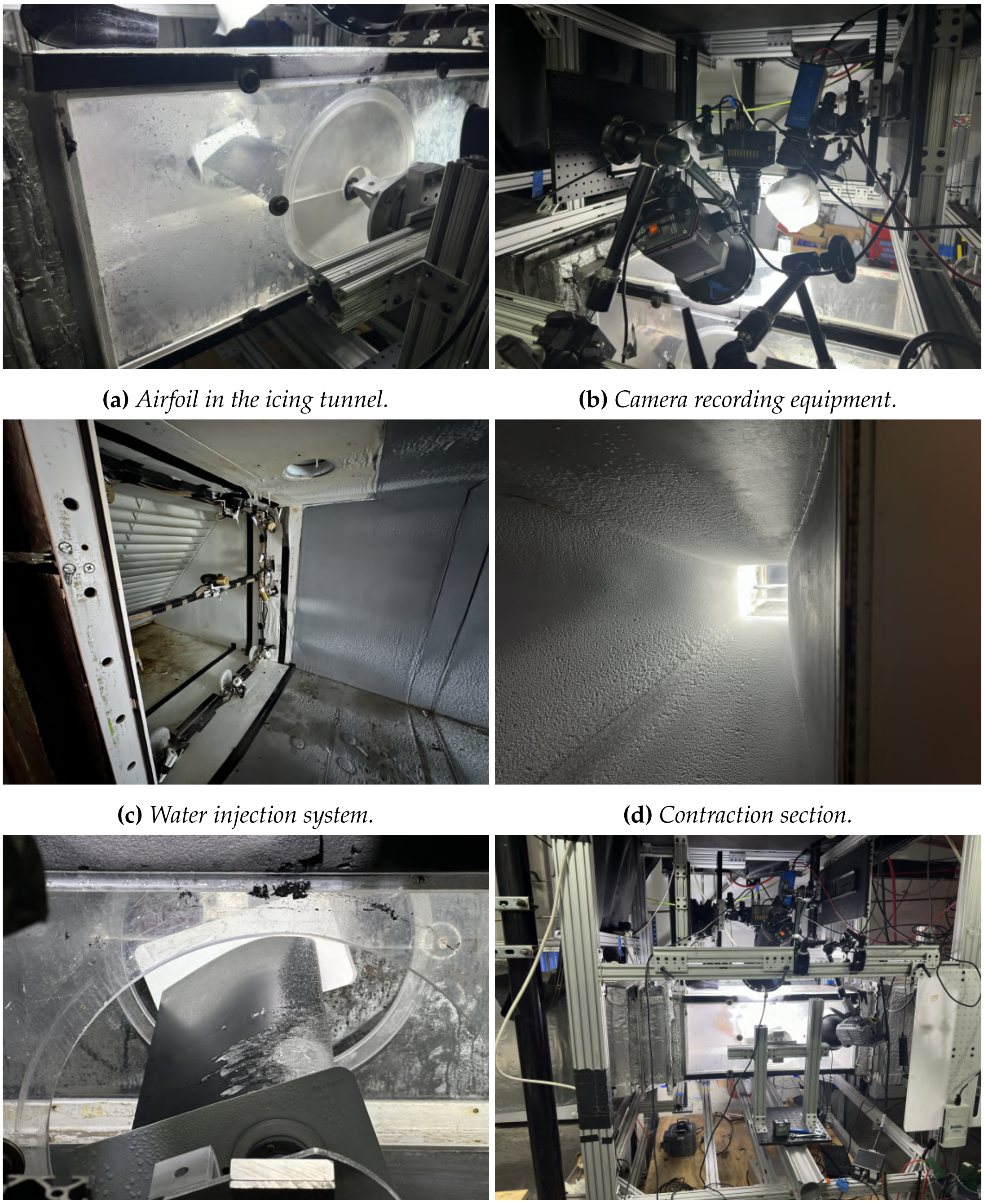
A system is set up to analyze the flow over an airfoil in a wind tunnel. This system begins with putting an airfoil in the test section, as seen in Figure 2.1a. Figure 2.1b shows the camera recording equipment that is in place above the test section. A water injector system, such as the one in Figure 2.1c is used to spray water into the tunnel, resulting in the formation of ice. Ice formation is noticeable in Figure 2.1d. This results in ice formation on the airfoil, seen in Figure 2.1e. Figure 2.1f depicts the complete setup, ready for operation.

2.2 Procedures

1. Turn the wind tunnel on.
2. Set the AoA of the airfoil to the first value in the data sheet.
3. Using the MATLAB application, record data for 10 s. Run this data in the provided processing script to generate values for c_l, c_d, and c_m.
4. Repeat steps 2 to 3 for all AoA in the data sheet.
5. Turn on the water spray system and wait for ice to accrue on the airfoil.
6. Repeat steps 2 to 3 for all AoA in the data sheet.

2.3 Derivations

The data in this lab was collected and processed using a provided MATrix LABoratory (MATLAB) script. This script outputted values for the coefficient of lift, c_l, coefficient of drag, c_d, and coefficient of moment, c_m. This data was plotted using the MATLAB script in Section C.1. The only calculation required was to calculate the lift to drag ratio, $\frac{c_l}{c_d}$. This ratio was determined by dividing the c_l by the c_d for each angle of attack.



(a) *Airfoil in the icing tunnel.*

(b) *Camera recording equipment.*

(c) *Water injection system.*

(d) *Contraction section.*

(e) *Close-up of an iced airfoil.*

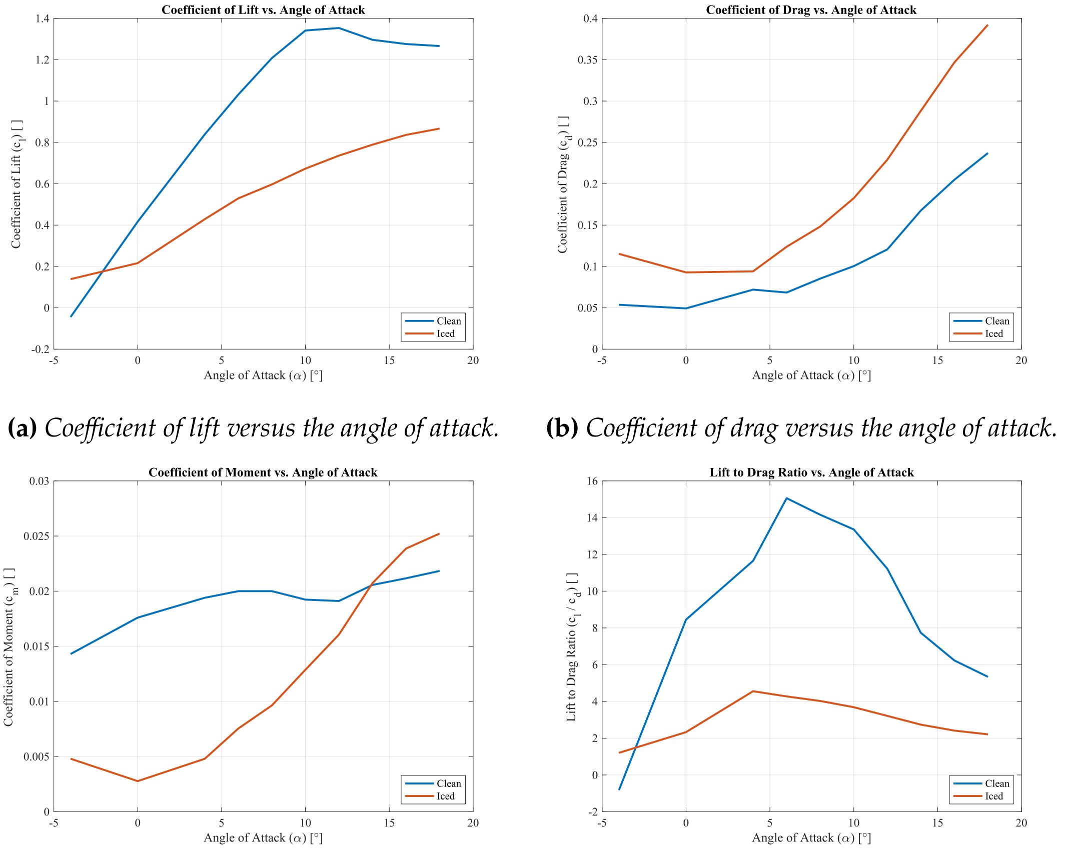
(f) *Full icing test apparatus.*

Figure 2.1: *Pictures of the icing wind tunnel apparatus.*

3

Results

Figure 3.1a, Figure 3.1b, and Figure 3.1c show the airfoil coefficient of lift, drag, and moment, respectively, for each angle of attack under clean and iced conditions. Figure 3.1d shows the airfoil lift to drag ratio for each angle of attack under clean and iced conditions. Full-size graphs can be found in Section B.2.



(a) *Coefficient of lift versus the angle of attack.* **(b)** *Coefficient of drag versus the angle of attack.*

(c) *Coefficient of moment versus the angle of attack.* **(d)** *Lift to drag ratio versus the angle of attack.*

Figure 3.1: *Graphs of various aerodynamic parameters for an airfoil under iced and clean conditions.*

4 Discussion

According to the graphs in Figure 3.1, the iced airfoil generated significantly less lift and significantly more drag, especially at higher angles of attack. The lift to drag ratio for the iced airfoil is worse (closer to zero), with the highest delta occurring from angles of attack 4° to 12°. This latter result is perfectly sensible, given that at high angles of attack, (above 12°) much of the flow is separating as it passes over the airfoil, hence less air is contacting the ice build-up on the airfoil.

As shown in Figure 3.1c, the coefficient of moment, c_m, is significantly lower on the iced airfoil at lower angles of attack. In steady flight, pilots will use the elevator to trim any excess moment caused by the wings. If a pilot were flying in trimmed flight and the wing suddenly iced over, it would cause the plane to pitch and the pilot would need to adjust the elevator to correct the trim. Additionally, if a pilot was flying in steady flight and the wings iced over, the pilot may also have to increase thrust to compensate for the lower $\frac{c_l}{c_d}$. Higher thrust necessarily means higher fuel usage. Furthermore, reducing the effects of icing could reduce fuel consumption expense.

It's hard to estimate or make any uncertainty measurements since we don't have information on the sensors or access to the script that ran the calculations. All sensors will only be precise to a degree, but taking a 10 s sample as we did in this lab will help to improve accuracy by smoothing out outliers and erroneous measurements.

5

Conclusion

To analyze the effects of airfoil icing on aerodynamic forces, we used the icing wind tunnel at Iowa State University to collect data on both a clean airfoil and an iced airfoil. From the data, we were able to find the coefficients of lift, drag, and moment and the lift to drag ratio at various AoA. Analyzing the various coefficients, we concluded that icing on the airfoil generally decreases lift, increases drag, and decreases the moment. For pilots flying in icy condition, this has two potential side effects: unwanted disturbances and higher required thrust to maintain steady flight. Finding ways to mitigate the effects of icing on aircraft not only saves money but may also save lives.

Bibliography

Hu, Hui (2024). *Aerodynamic Force Measurement on an Icing Airfoil*. Iowa State University. URL: https://www.aere.iastate.edu/~huhui/teaching/2024-01S/AerE344/lab-instruction/AerE344L-Lab-12-instruction.pdf.

A

Appendix A

A.1 Data Tables

Table A.1: *Measured aerodynamic characteristics for a clean airfoil where α is the angle of attack, c_l is the coefficient of lift, c_d is the coefficient of drag, c_m is the coefficient of moment, and $\frac{c_l}{c_d}$ is the lift to drag ratio.*

α [°]	c_l	c_d	c_m	$\frac{c_l}{c_d}$
−4.0	−0.0445	0.0537	0.0143	−0.829
0.0	0.417	0.0493	0.0176	8.45
4.0	0.839	0.0720	0.0194	11.7
6.0	1.03	0.0685	0.0200	15.1
8.0	1.21	0.0853	0.0200	14.2
10	1.34	0.100	0.0192	13.4
12	1.35	0.121	0.0191	11.2
14	1.30	0.168	0.0206	7.73
16	1.28	0.205	0.0212	6.23
18	1.27	0.237	0.0218	5.34

Table A.2: *Measured aerodynamic characteristics for an iced airfoil where α is the angle of attack, c_l is the coefficient of lift, c_d is the coefficient of drag, c_m is the coefficient of moment, and $\frac{c_l}{c_d}$ is the lift to drag ratio.*

α [°]	c_l	c_d	c_m	$\frac{c_l}{c_d}$
−4.0	0.139	0.115	4.80×10^{-3}	1.20
0.0	0.216	0.0928	2.77×10^{-3}	2.33
4.0	0.429	0.0941	4.80×10^{-3}	4.56
6.0	0.529	0.124	7.54×10^{-3}	4.27
8.0	0.597	0.148	9.64×10^{-3}	4.02
10	0.673	0.183	0.0129	3.69
12	0.736	0.229	0.0161	3.21
14	0.789	0.288	0.0207	2.74
16	0.836	0.347	0.0239	2.41
18	0.867	0.392	0.0252	2.21

B
Appendix B

B.1 Additional Apparatus Pictures

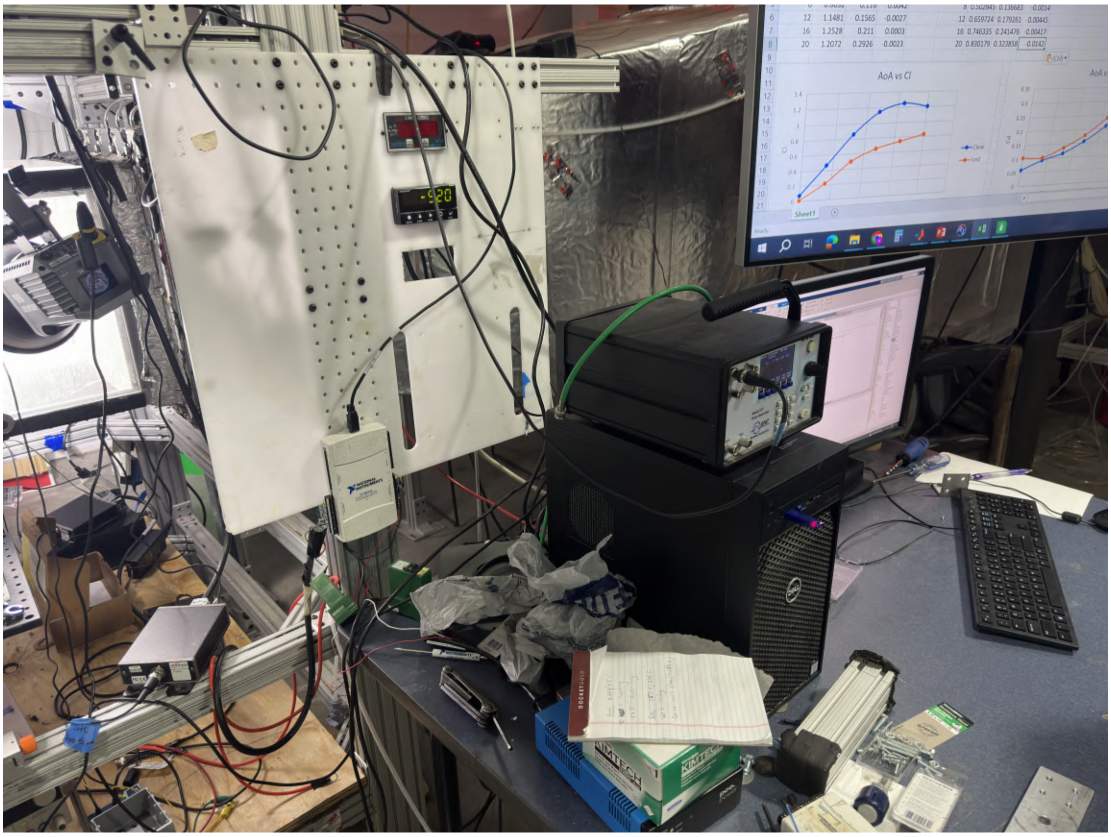


Figure B.1: *Data collection computer and equipment.*

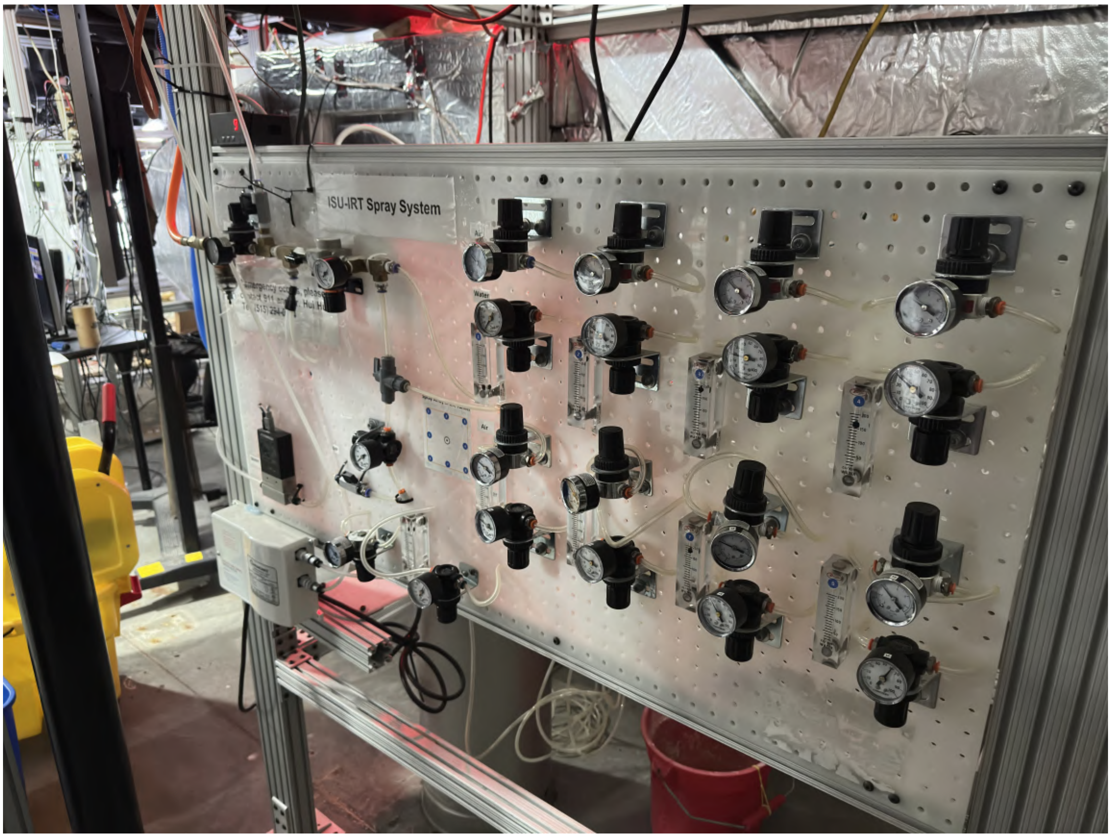


Figure B.2: *Spray nozzle control panel.*

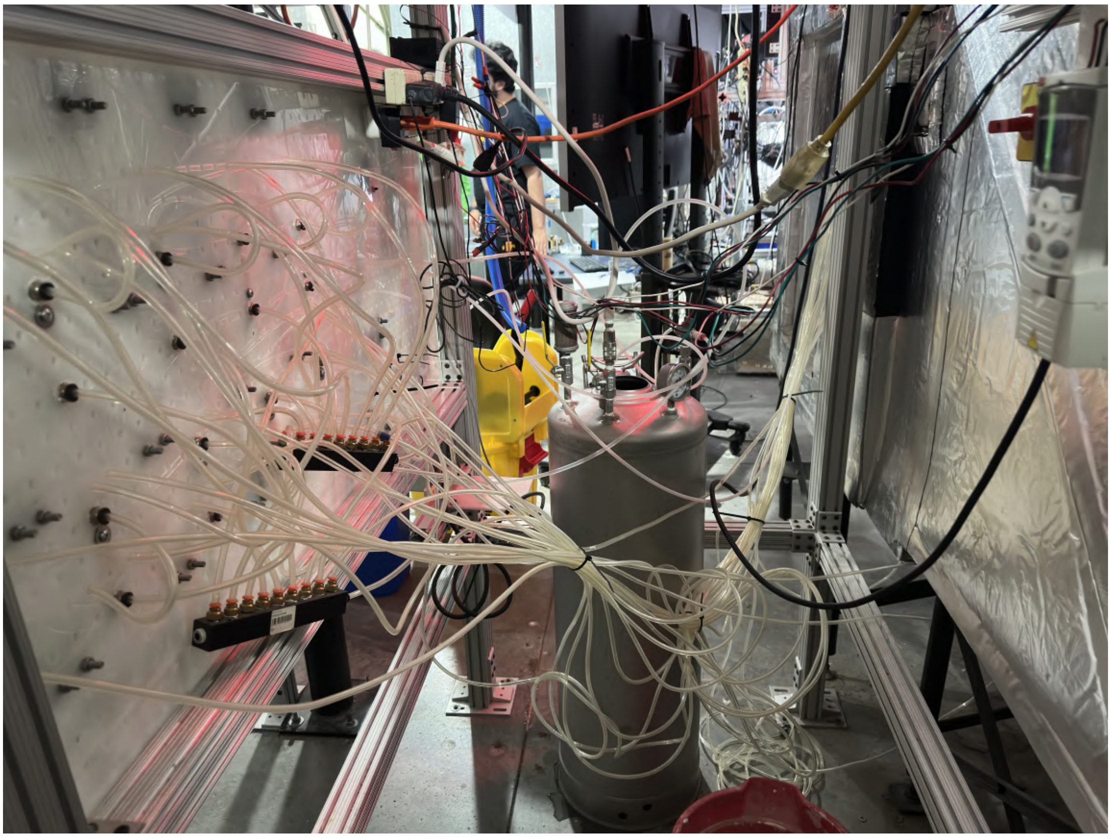


Figure B.3: *Water tubes for the spray nozzle system.*

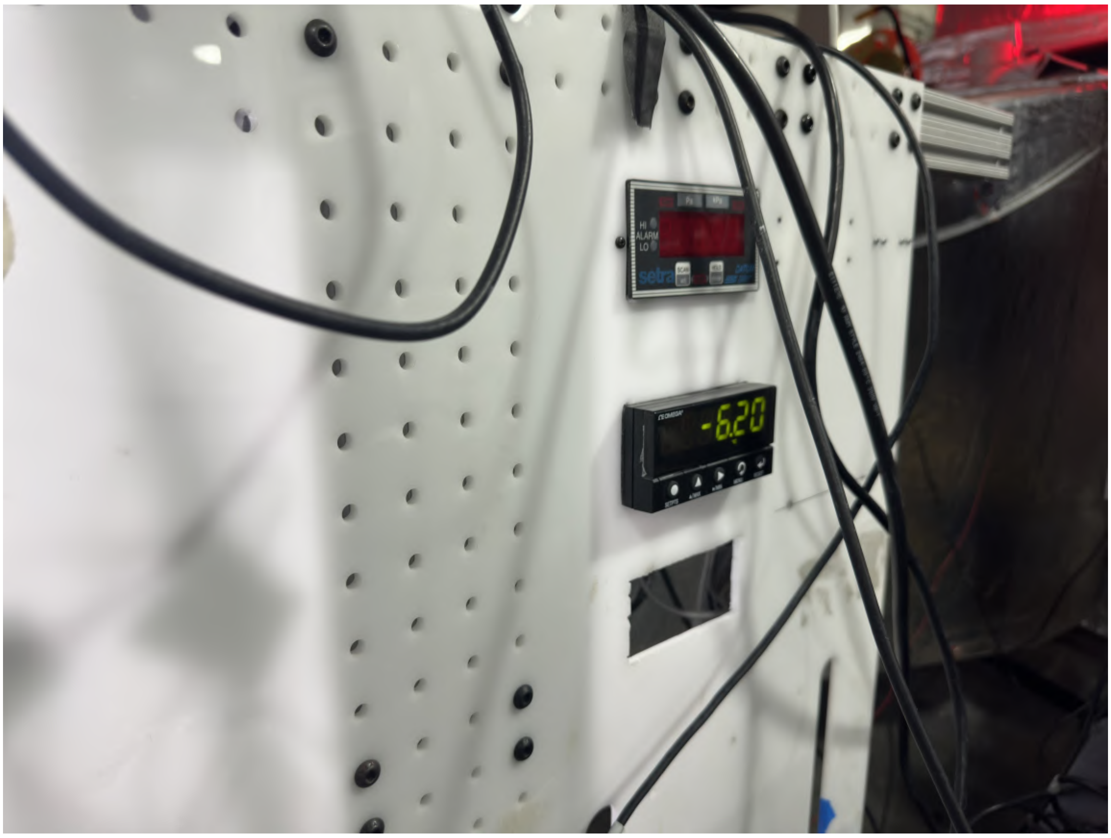


Figure B.4: *Wind tunnel temperature gauge.*

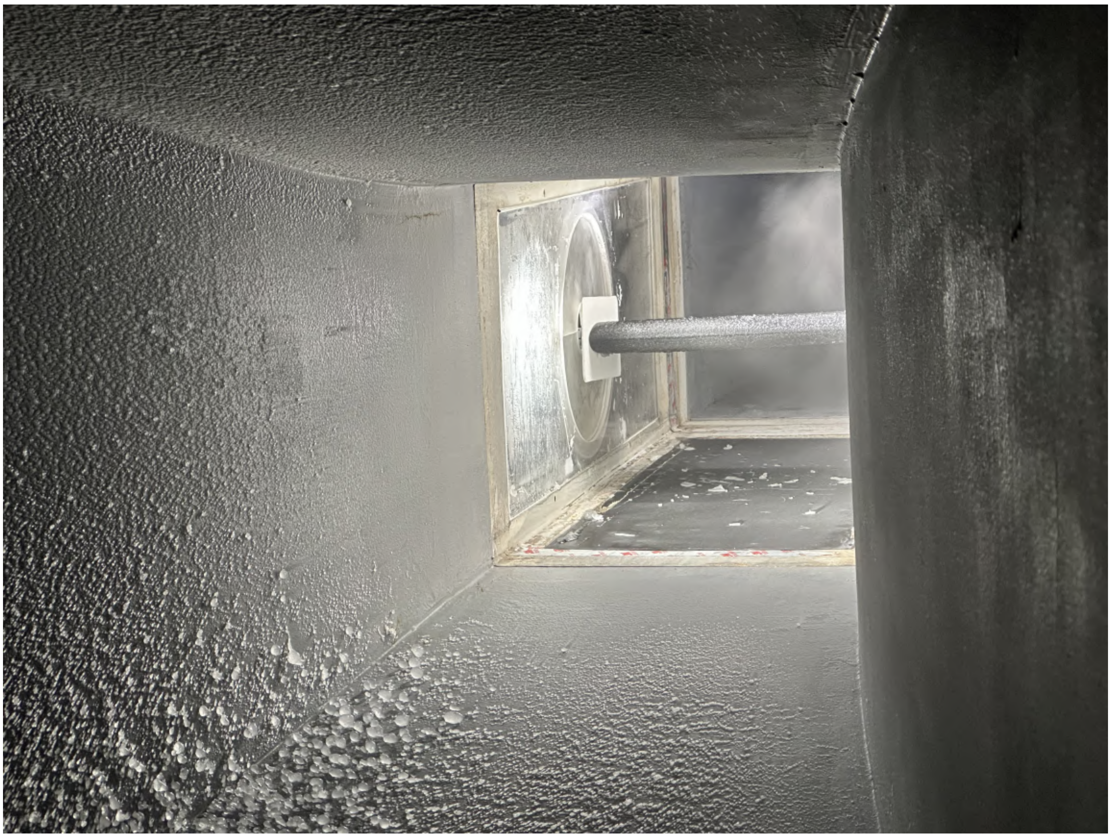


Figure B.5: *Start of the test section.*

B.2 Full Size Figures

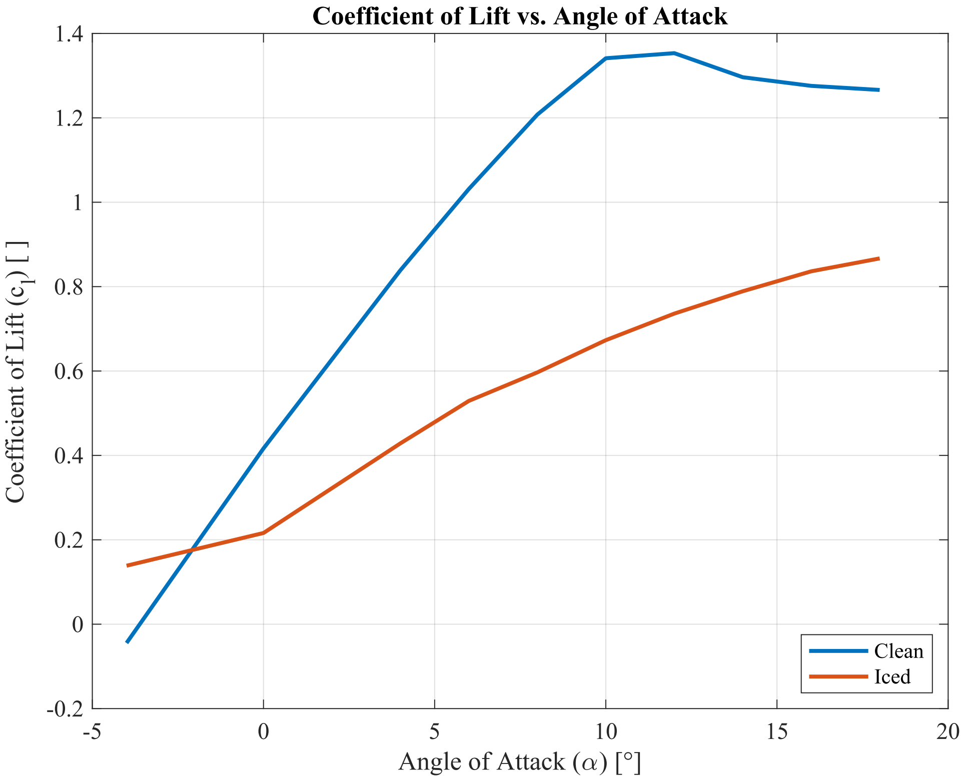


Figure B.6: *Full size graph of the coefficient of lift versus the angle of attack.*

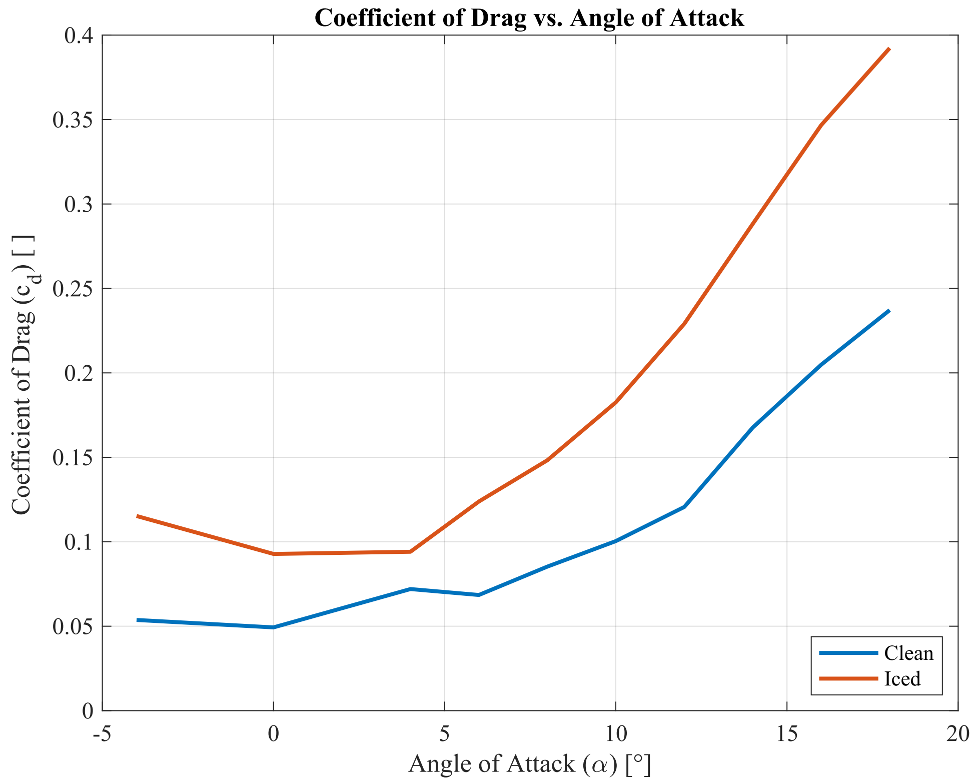


Figure B.7: *Full size graph of the coefficient of drag versus the angle of attack.*

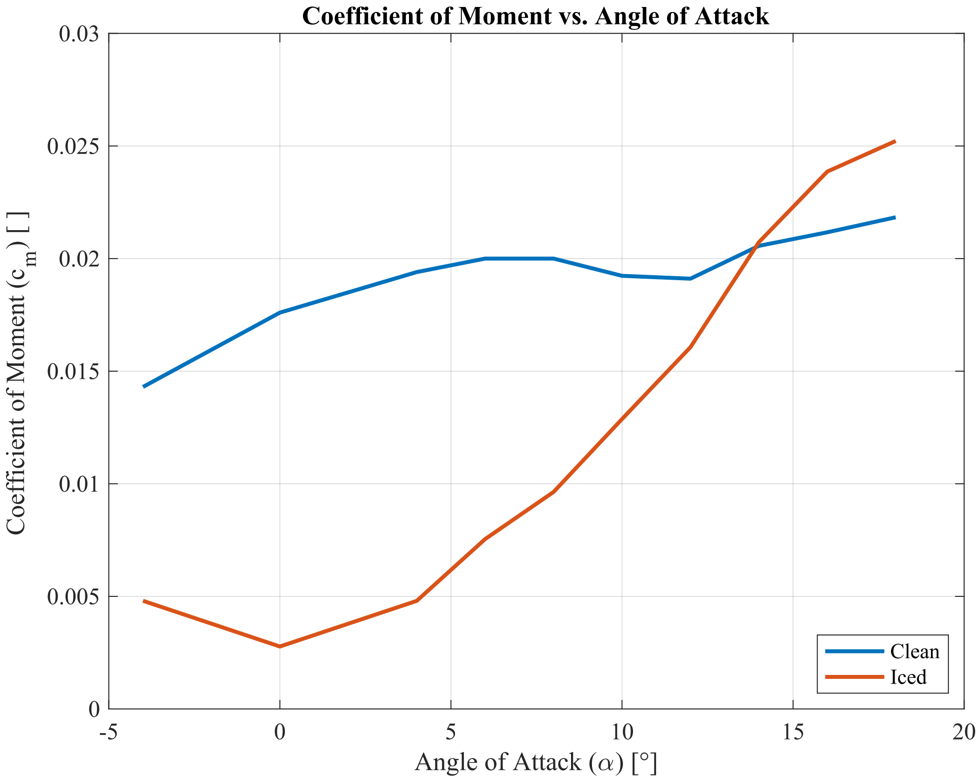


Figure B.8: *Full size graph of the coefficient of moment versus the angle of attack.*

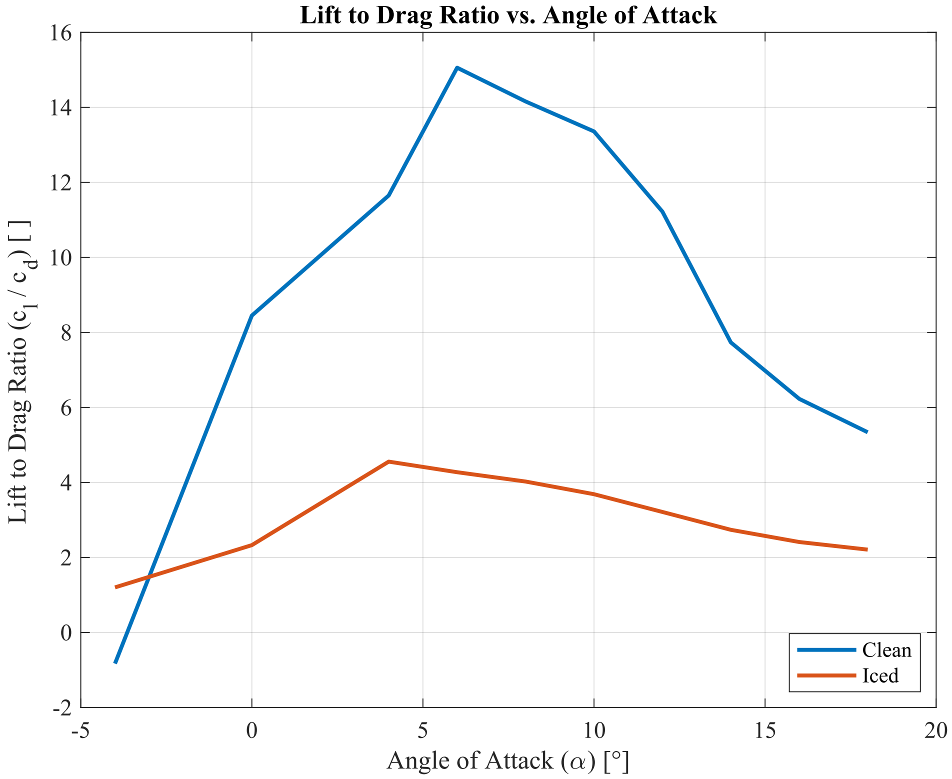


Figure B.9: *Full size graph of the lift to drag ratio versus the angle of attack.*

C

Appendix C

C.1 lab12analysis.m

```
% Spring 2024 AER E 344 Lab 12 Analysis Script
% Section 3 Group 3
clear; clc; close all;

%% Constants
data_file = "./AERE344Lab12Data.xlsx";
figure_dir = "../Figures/";

%% Import Data
data = readtable(data_file);
alpha = data.AoA; % [°]
c_l = data.cl; % []
c_d = data.cd; % []
c_m = data.cm; % []
c_l_iced = data.cl_1; % []
c_d_iced = data.cd_1; % []
c_m_iced = data.cm_1; % []

%% Calculations
c_lbyc_d = c_l ./ c_d; % []
c_lbyc_d_iced = c_l_iced ./ c_d_iced; % []

%% Plot
figure;
plot(alpha,c_l,"LineWidth",2);
hold on;
plot(alpha,c_l_iced,"LineWidth",2);
hold off;
fontname("Times New Roman");
fontsize(12,"points");
title_str = "Coefficient of Lift vs. Angle of Attack";
title(title_str);
xlabel("Angle of Attack (\alpha) [°]");
ylabel("Coefficient of Lift (c_l) [ ]");
```

```
legend("Clean","Iced","Location","southeast");
grid on;
saveas(gcf,figure_dir + title_str + ".svg");

figure;
plot(alpha,c_d,"LineWidth",2);
hold on;
plot(alpha,c_d_iced,"LineWidth",2);
hold off;
fontname("Times New Roman");
fontsize(12,"points");
title_str = "Coefficient of Drag vs. Angle of Attack";
title(title_str);
xlabel("Angle of Attack (\alpha) [°]");
ylabel("Coefficient of Drag (c_d) [ ]");
legend("Clean","Iced","Location","southeast");
grid on;
saveas(gcf,figure_dir + title_str + ".svg");

figure;
plot(alpha,c_m,"LineWidth",2);
hold on;
plot(alpha,c_m_iced,"LineWidth",2);
hold off;
fontname("Times New Roman");
fontsize(12,"points");
title_str = "Coefficient of Moment vs. Angle of Attack";
title(title_str);
xlabel("Angle of Attack (\alpha) [°]");
ylabel("Coefficient of Moment (c_m) [ ]");
legend("Clean","Iced","Location","southeast");
grid on;
saveas(gcf,figure_dir + title_str + ".svg");

figure;
plot(alpha,c_lbyc_d,"LineWidth",2);
hold on;
plot(alpha,c_lbyc_d_iced,"LineWidth",2);
hold off;
fontname("Times New Roman");
fontsize(12,"points");
title_str = "Lift to Drag Ratio vs. Angle of Attack";
title(title_str);
xlabel("Angle of Attack (\alpha) [°]");
ylabel("Lift to Drag Ratio (c_l / c_d) [ ]");
legend("Clean","Iced","Location","southeast");
grid on;
saveas(gcf,figure_dir + title_str + ".svg");

%% Output Tables
clean_table = table;
```

```
clean_table.alpha = alpha;
clean_table.c_l = c_l;
clean_table.c_d = c_d;
clean_table.c_m = c_m;
clean_table.c_lbyc_d = c_lbyc_d;
path = convertStringsToChars(figure_dir + "Clean Data.tex");
table2latex(clean_table,path, { ...
    '$\alpha$ [\unit{\degree}]', ...
    '$c_l$', ...
    '$c_d$', ...
    '$c_m$', ...
    '$\frac{c_l}{c_d}$' ...
    },[2,3,3,3,3],[]);

iced_table = table;
iced_table.alpha = alpha;
iced_table.c_l = c_l_iced;
iced_table.c_d = c_d_iced;
iced_table.c_m = c_m_iced;
iced_table.c_lbyc_d = c_lbyc_d_iced;
path = convertStringsToChars(figure_dir + "Iced Data.tex");
table2latex(iced_table,path, { ...
    '$\alpha$ [\unit{\degree}]', ...
    '$c_l$', ...
    '$c_d$', ...
    '$c_m$', ...
    '$\frac{c_l}{c_d}$' ...
    },[2,3,3,3,3],[]);
```

C.2 table2latex.m

```
function table2latex(T, filename, column_names, sigfigs, ignore_col)
    if nargin < 2
        filename = 'table.tex';
        fprintf(['Output path is not defined. The table will be ' ...
            'written in %s.\n'], filename);
    elseif ~ischar(filename)
        error('The output file name must be a string.');
    else
        if ~strcmp(filename(end-3:end), '.tex')
            filename = [filename '.tex'];
        end
    end
    if nargin < 1, error('Not enough parameters.'); end
    if ~istable(T), error('Input must be a table.'); end

    % Parameters
    n_col = size(T,2);
    col_spec = [];
    for c = 1:n_col, col_spec = [col_spec 'c']; end
    col_names = strjoin(column_names, ' & ');
    row_names = T.Properties.RowNames;
    if ~isempty(row_names)
        col_spec = ['l' col_spec];
        col_names = ['& ' col_names];
    end

    % Writing header
    fileID = fopen(filename, 'w');
    fprintf(fileID, '\\begin{tabular}{%s}\n', col_spec);
    fprintf(fileID, '\\toprule\n');
    fprintf(fileID, '%s \\\\\n', col_names);
    fprintf(fileID, '\\midrule\n');

    % Writing the data
    for row = 1:size(T,1)
        temp{1,n_col} = [];
        for col = 1:n_col
            value = T{row,col};
            if isstruct(value)
                error('Table must not contain structs.');
            end
            while iscell(value), value = value{1,1}; end
            if isinf(value), value = '$\infty$'; end
            if ismember(col,ignore_col)
                temp{1,col} = '';
            else
                temp{1,col} = convertStringsToChars("\num{" ...
                    + sigfig(value,sigfigs(col)) + "}");
            end
        end
        if ~isempty(row_names)
            temp = [row_names{row}, temp];
        end
        fprintf(fileID, '%s \\\\ \n', strjoin(temp, ' & '));
        clear temp;
    end

    % Closing the file
    fprintf(fileID, '\\bottomrule\n');
    fprintf(fileID, '\\end{tabular}');
    fclose(fileID);
end
```

C.3 sigfig.m

```
%[strOut2] = sigfig(matNum, nSigFig, strPad)
%Rounds number to nSigFig number of significant figures and outputs a string
%'pad' in 3rd argument to have padded zeros, else unpadded
%if number of arguments < 3, then choose shorter output, between padded and unpadded
%if number of arguments < 2, then 3 significant figures
%Lim Teck Por, 2006, 2008, 2009
%Apropos: mat2str, num2str, sprintf

function [strOut2] = sigfig(matNum, nSigFig, strPad)
[N, D] = size(matNum);
if (nargin < 2)
    nSigFig = 3;
end
if (nargin < 3)
    strPad = [];
end

strOut2 = [];
for l = 1:N
    for k = 1:D
        numkl = matNum(l,k);
        if (isnan(numkl)||isinf(numkl)) %if nan or inf
            strOut = num2str(numkl);
            mySign = [];
        else %if neither nan or inf
            if (sign(numkl) == -1)
                mySign = '-';
            else
                mySign = [];
            end
            num = abs(numkl);
            nSigFig1 = nSigFig - 1;
            strFormat = ['%1.',(num2str(nSigFig+2)),'e'];

            strTemp = sprintf(strFormat, num);
            [strPrefix,strExponent] = strtok(strTemp, 'e');
            strExponent = strExponent(2:end);
            strFactor = num2str(nSigFig1);
            nTemp = str2num([strPrefix, 'e', strFactor]);
            nExponent = str2num(strExponent);
            fTemp = str2num([num2str(round(nTemp)), 'e', num2str(nExponent-nSigFig1)]);

            strTemp = sprintf(strFormat, fTemp);
            [strPrefix,strExponent] = strtok(strTemp, 'e');
            strExponent = strExponent(2:end);
            while (strExponent(2) == '0') && (length(strExponent) > 2)
                strExponent = [strExponent(1), strExponent(3:end)];
            end
            [strPrefix2,strSuffix2] = strtok(strPrefix, '.');
            strSuffix2 = strSuffix2(2:end);
            if (str2num(strSuffix2(nSigFig)) >= 5)
                nTemp = str2num([strPrefix2,strSuffix2(1:nSigFig1)])+1;
                strTemp2 = num2str(nTemp);
                strPrefix2 = strTemp2(1);
                strSuffix2 = strTemp2(2:end);
            else
                strSuffix2(nSigFig:end) = [];
            end
            if (nargin < 3) %if zero padding
                strOuta = zeroPadding(strPrefix2, strSuffix2, strExponent, nSigFig,
                ↪  num, strPad);
                if (nSigFig1 == 0)
                    strOutb = [strPrefix2, strSuffix2, 'e', strExponent];
                else
                    strOutb = [strPrefix2, '.', strSuffix2, 'e', strExponent];
                end
                if(length(strOuta)<length(strOutb))
                    strOut = strOuta;
                else
                    strOut = strOutb;
                end
            else %if no zero padding
                if (strcmp(strPad,'pad'))
                    strOut = zeroPadding(strPrefix2, strSuffix2, strExponent, nSigFig,
                    ↪  num, strPad);
                else
                    if (nSigFig1 == 0)
                        strOut = [strPrefix2, strSuffix2, 'e', strExponent];
                    else
                        strOut = [strPrefix2, '.', strSuffix2, 'e', strExponent];
                    end
                end
            end %if no zero padding
            if (strOut(end)=='.')
                strOut = strOut(1:end-1);
            end
            if (length(strOut) > 5)
                if (strcmpi(strOut(end-2:end), 'e+0'))
                    strOut = strOut(1:end-3);
                end
            end
        end %if neither nan or inf
        strOut2 = [strOut2, mySign, strOut];
        if (k<D)
            strOut2 = [strOut2, ','];
        end
    end
    if (l<N)
        strOut2 = [strOut2, ';'];
    else
        strOut2 = sprintf('%s', strOut2);
    end
end

function [strOut] = zeroPadding(strPrefix2, strSuffix2, strExponent, nSigFig, num, strPad)
nDP = str2num(strExponent);
if (nDP < 0) %nDP < 0
    strZeros = char(repmat(48,1,abs(nDP)-1));
    strOut = ['0.', strZeros, strPrefix2, strSuffix2];
else %nDP >= 0
    nP = length(strPrefix2);
    nS = length(strSuffix2);
    nPad =  nSigFig - nP - nS;
    if (nPad > 0)
        strZeros = char(repmat(48,1,nPad));
    else
        strZeros = [];
    end
    if (nDP == 0) %nDP = 0
        strOut = [strPrefix2, '.', strSuffix2, strZeros];
    else %nDP > 0
        %nOut = str2num([strPrefix2, '.', strSuffix2]);
        %strOut = num2str(nOut*10^nDP);
        nPad1 = nDP - nS;
        strZeros1 = char(repmat(48,1,nPad1));
        strTemp = [strSuffix2, strZeros1];
        strOut = [strPrefix2, strTemp(1:nDP), '.', strTemp(nDP+1:end)];
        nPad2 = nSigFig - length(strOut);
        if (nPad2 > 0)
            strZeros2 = char(repmat(48,1,nPad2));
            strOut = [strOut, '.', strZeros2];
        end
    end %nDP > 0
end %nDP >= 0
```