Table of Contents

ASEN 3111 - Computational Assignment 1 - Main

```
%{

This is the main script to answer the deliverables for Computational
Assignment 1 (CA 1), you can find the deliverables in the folder /
info.

This script will use other routines in the folder Scripts/


Author: Abdulla Al Ameri
Collaborators: Brendan Palmer, Abdullah Almugairin, Samuel D'Souza
Date: Sept, 2nd, 2019.

%}
```

house keeping

```
clear
clc
close all
```

add scripts folder to path

```
% the ./ means all paths up to the current folder that contains
 Main.m,
% then in this path there's a folder called Scripts, add that to us.


addpath('./Scripts');
```

question 1:

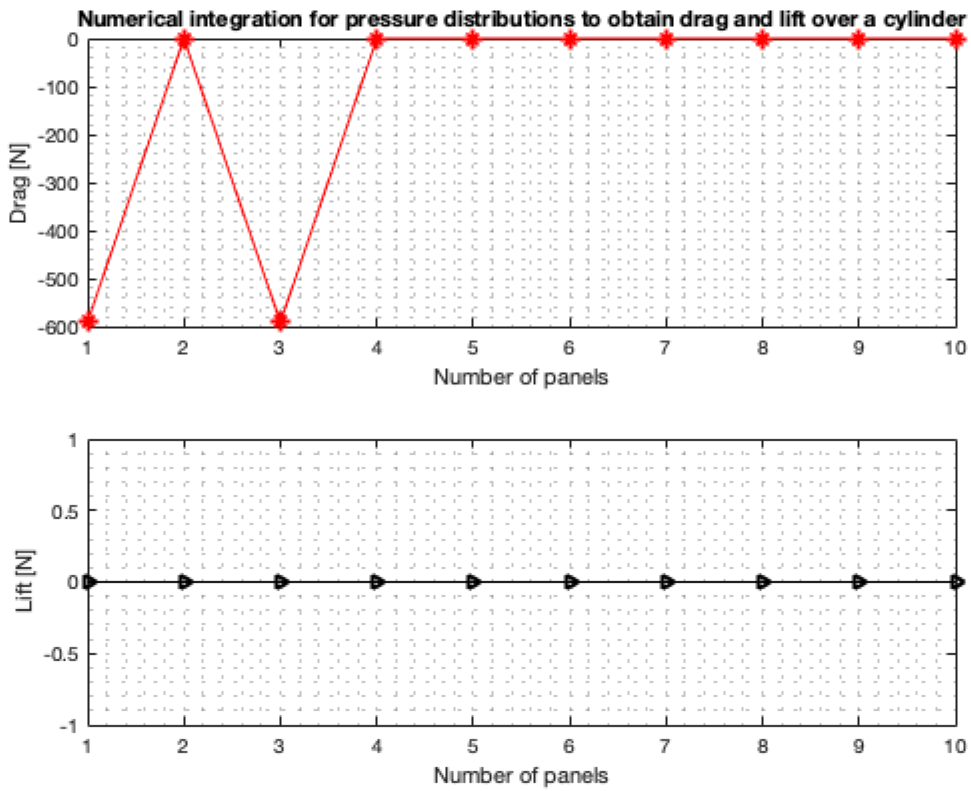
call scripts related to question 1

```
run('Q1.m');
```

```
-=-=-=-=-=-=( Question 1 ) -=-=-=-=-=-=-=-=-=-=-=-=-

Elapsed time is 2.720297 seconds.
Number of panels for Drag to be within 0.001N is: 3
Number of panels for Lift to be within 0.001N is: 1

-=-=-=-=-=-=( END ) -=-=-=-=-=-=-=-=-=-=-=-=-
```



question 2:

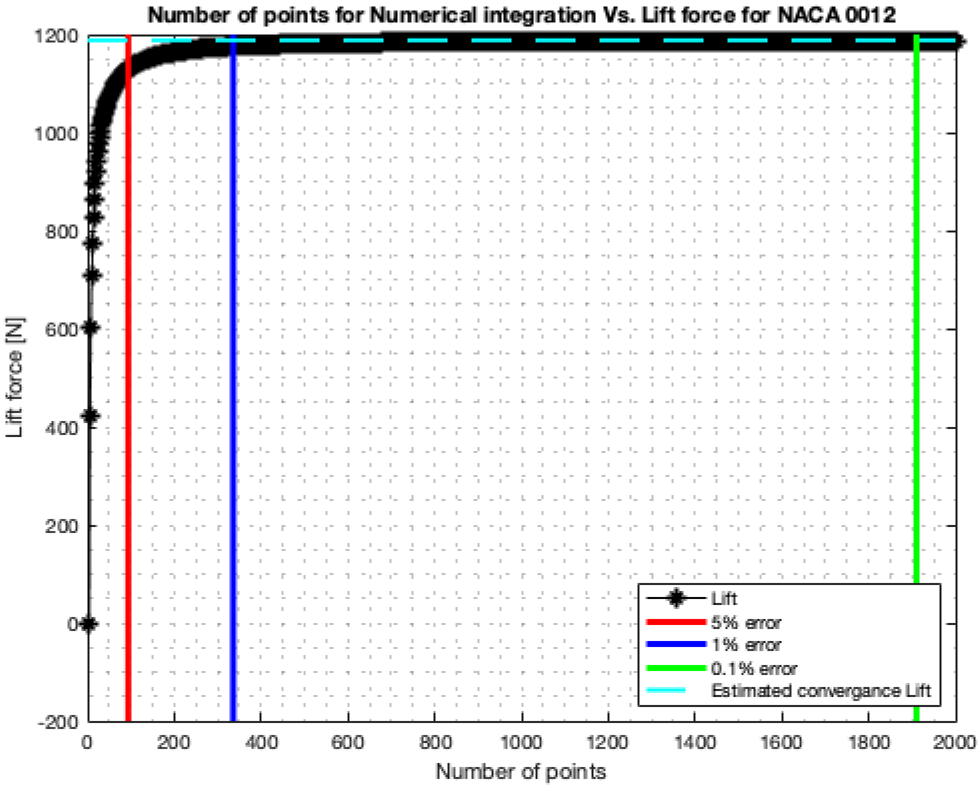
call scripts related to question 2

```
run('Q2.m');
```

```
-=-=-=-=-=-=( Question 2 ) -=-=-=-=-=-=-=-=-=-=-=-=-

Elapsed time is 0.899613 seconds.
Number of integration points n for L to be within 5%  relative error
 is: 94
Number of integration points n for L to be within 1%  relative error
 is: 336
Number of integration points n for L to be within 0.1%  relative error
 is: 1910
```

```
-=-=-=-=-=-=( END ) -=-=-=-=-=-=-=-=-=-=-=-
```



functions used

```
%% question 1: estimate left and drage over a cylinder.


%{

cylinder is at 0 angle of attack (AOA). Therfore, we can assume
that Lift = Normal force, Drag = Axial force, for the flow stream
velocity alligns with the chors.

A direct integration of the coefficients of pressure will hence
give us the coefficients of normal(lift) and axial(drag) forces.


It can be observed from figure in problem doc (go to /info) that
an integration over the circle from 0 to 2*pi will give us the
coefficients. Since Cp is given as function of theta, we can integrate
```

```
directly. Cp here can represent the vector field, the surface is the
circle.

%}

%% housekeeping

clear
clc
close all

tic

%% define constants

d = 1 ; % daiameter [m]
roh_inf = 1.255 ; % free-stream density [kg/m^3]
p_inf = 101.3*10^3 ; % static pressure [Pa]
V_inf = 30 ; % free-stream velocity [m/s]
q_inf = 1/2 * roh_inf * V_inf^2 ; % dynamic pressure [Pa]
c = d ; % chord length here = diameter [m].


%% solve: analytical
% the analytical slloution:

% we can see that since Cp will always be alligned radially with the
 center
% of the circle, that Cp*cos(theta) = drag, Cp*sin(theta) = lift.

syms th

%% QUESTIONS:

% IS this below wrong trating Cp as Pressures? should we have instead
 done the
% method explained in 1.15, 1.16?

%%

Cp = 1 - 4*(sin(th)^2);
Cl = Cp * sin(th);
Cd = Cp * cos(th);


% integrate symbolically:
Analytical_int_Cl = (1/c)*(d/2)*(int(Cl,[0 2*pi])); % this 1/c in the
 beggining is important to make the resultant non-dimensional.
Analytical_int_Cd = (1/c)*(d/2)*(int(Cd,[0 2*pi]));


% this analytical slloution can be used to quantify the error.

Analytical_L =  Analytical_int_Cl * (1/2) * roh_inf * V_inf^2 * (c);
```

```
Analytical_D =  Analytical_int_Cd * (1/2) * roh_inf * V_inf^2 * (c);

%% solve: numerical

% the following is just for testing:

% using Simpson's rule.
Numerical_int_Cl = (1/c)*(d/2)*CompositeSimpsons(Cl,10,0,2*pi);
Numerical_int_Cd = (1/c)*(d/2)*CompositeSimpsons(Cd,10,0,2*pi);

% d/2 is raduis, pulled out of series because it's constant.

% estimate Lift and Drag
Numerical_D = Numerical_int_Cd * (1/2) * roh_inf * V_inf^2 * (c);
Numerical_L = Numerical_int_Cl * (1/2) * roh_inf * V_inf^2 * (c);


%% see how accurate is numerical integration:
j = 1;
for i = 1:1:10

Numerical_int_Cl_Different_N(j) = (1/
c)*(d/2)*CompositeSimpsons(Cl,i,0,2*pi);
Numerical_int_Cd_Different_N(j) = (1/
c)*(d/2)*CompositeSimpsons(Cd,i,0,2*pi);
Different_N(j) = i; % store number of panels
    j = j+1;

end

Numerical_D_Different_N =  Numerical_int_Cd_Different_N .* (1/2) *
 roh_inf * V_inf^2 * (d) ;
Numerical_L_Different_N = Numerical_int_Cl_Different_N .* (1/2) *
 roh_inf * V_inf^2 * (d) ;



figure(1)
subplot(2,1,1)
plot(Different_N,Numerical_D_Different_N,'r-*')
xlabel('Number of panels');
ylabel('Drag [N]')
grid minor
title('Numerical integration for pressure distributions to obtain drag
 and lift over a cylinder')

subplot(2,1,2)
plot(Different_N,Numerical_L_Different_N,'k->')
xlabel('Number of panels');
ylabel('Lift [N]')
grid minor

%% relative error:
```

```
% relative error would be taken relative to analytical sloution.

Error_L = double(abs(Numerical_L_Different_N - Analytical_L )) ;
Error_D = double(abs(Numerical_D_Different_N - Analytical_D )) ;

%% determine number of panels needed:

Tolerance = 0.001; % tolerance we need to be within;

Condition_Error_D = find(diff(Error_D<Tolerance)==1);

if isempty(Condition_Error_D) == 1
    % if it converges immeaditly then pick number of panels == 1;
    Condition_Error_D = 1;
else
    % if it doesn't, see where it does.
    Condition_Error_D = Condition_Error_D(end);
end


% check when lift converges


Condition_Error_L = find(diff(Error_L<Tolerance)==1);

if isempty(Condition_Error_L) == 1
    Condition_Error_L = 1;

else
    Condition_Error_L = Condition_Error_L(end);
end



%% ptrinout results

fprintf('-=-=-=-=-=-=( Question 1 ) -=-=-=-=-=-=-=-=-=-=-=-=-');
fprintf('\n \n')

toc

% Printout results
fprintf(['Number of panels for Drag to
 be within ' num2str(Tolerance) 'N is: '
 num2str(Different_N(Condition_Error_D))]);
fprintf('\n')

fprintf(['Number of panels for Lift to
 be within ' num2str(Tolerance) 'N is: '
 num2str(Different_N(Condition_Error_L))]);
fprintf('\n \n')

fprintf('-=-=-=-=-=-=( END ) -=-=-=-=-=-=-=-=-=-=-=-=-');
fprintf('\n \n \n')
```

functions used

```
function [ int ] = CompositeSimpsons(f,N,a,b)
% This function will perform numerical line integration using
% composite simpsons rule, the user will pass in parameterized
% vector field function f, and number of segments (panels) the code
% will return the result of the definite integral.
%
% - - - - - - - - - - - - - - - - - - - - - - - - - -
%
%       Inputs:
%               1- f: parameterized vector field function
%               2- N: number of segments (panels)
%               3- a: Lower bound of integration
%               4- b: Upper bound of integration
%
% - - - - - - - - - - - - - - - - - - - - - - - - - -
%
%       Outputs:
%               1- int: result of definite integral
% - - - - - - - - - - - - - - - - - - - - - - - - - -
%
%
%
%   -Abdulla AlAmeri
%   -CU Boulder, Fall 2019, ASEN 3111.

% define h:

h = ( b - a ) / (2*N);

% intiate results
int = 0;

for k = 1:N;

    x1 = a + ( (2*k -1) -1 ) * h;
    x2 = a + ( (2*k) -1 ) * h;
    x3 = a + ( (2*k +1) -1 ) * h;
    % calculate this itteration of the series
    series = subs(f,x1) + ...
        4*subs(f,x2) + ...
        subs(f,x3) ;

    % add new results to previous results
    int = int + series ;


end
```

```
% now multiply by the constants outside the series.
int = (h/3) * int ;
int = double(int);
```

functions used

```
%% question 2: NACA 0012

% you're given information about Cp disturbution NACA 0012 lower and
 upper
% surface estimate lift and drag using numerical integraiton,
% mainly Trapazoidal rule

%% housekeeping

clear



%% define ocnstants

tic

c = 2; % chord length [m]
alpha = 9; % AOA [Degrees]
V_inf = 30; % Free-stream flow speed [m/s]
roh_inf = 1.225; % Free-stream flow density [kg/m];
P_inf = 101.3*10^3 ; % free-stream pressure (statics)[Pa];


%% Airfoil information: NACA 0012

% integration the Coefficient of Axial force using equation 1.16 will
% require you to know the change in height of the airfoil over the
 change
% of the horizontal distance of the airfoil, therefore we need to
 define
% the thickness or yt of the airfoil:


% NACA Naming follows the following convention:

% NACA MPXX
%   example : NACA 2412
%     M is the maximum camber divided by 100. In the example M=2 so
 the camber is 0.02 or 2% of the chord
%     P is the position of the maximum camber divided by 10. In the
 example P=4 so the maximum camber is at 0.4 or 40% of the chord.
```

```
%     XX is the thickness divided by 100. In the example XX=12 so the
 thiickness is 0.12 or 12% of the chord.


% more info:

% http://airfoiltools.com/airfoil/naca4digit


% since it's NACA 0012 %thickness is t = 12/100

% In our example, M = 0; Therfore, gradient change is 0.
t = 12/100;

% since our airfoil isn't cambered, we can say that from equations in
 the
% website:

% since it's symmetric airfoil, we will see that everything cancels
 and the
% y location simply becomes

yt = @(x) (t/0.2).*c .* ( 0.2969.*sqrt(x./c) -0.1260*(x./c) -
 0.3516.*(x./c).^2 + 0.2843.*(x./c).^3 - 0.1036.*(x./c).^4 ) ;


Yu = yt;
Yl = @(x) (-1)*(t/0.2).*c .* ( 0.2969.*sqrt(x./c) -0.1260*(x./c) -
 0.3516.*(x./c).^2 + 0.2843.*(x./c).^3 - 0.1036.*(x./c).^4 ) ;


% for Ca we would need to know how the lower and upper surface change
% curvature, so we would need the derivative with respect to x.



%% info

% since it was given that the airfoil is at 9-degrees AOA, we know
 that
% integration over the surface will give us normal and axial forces,
 and
% those can be converted to lift and drag using AOA.


% using equation 1.15, 1.16 from Anderson's Fundemental of
 aerodynamics,
% page 26 in 6th edition:

% TrapezoidalRule will give out results from definite integral,
 there's no
% need to concern ourselves with skin friction pressure for now.

%% integration: to estimate the answers it converges to
```

```
% open up the spline function
Spline = open('Cp.mat');

% get upper and lower spline data
Cp_upper = Spline.Cp_upper;
Cp_lower = Spline.Cp_lower;


NumPoints = 10000; % how many panels used to integrate?

lower_limit = 0;
x = linspace(lower_limit,c,NumPoints); % create segments that we will
 integrate along


% important: getting to Cp's must be through x/c not x !
Cpu = fnval(Cp_upper, x./c); % evaluate the upper surface Cp's along
 that segment
Cpl = fnval(Cp_lower, x./c); % evaluate the lower surface Cp's along
 that segment


% since dx is fixed because points in x are equispaced, the change in
 y

Dyu= double(Yu(x));% evaluate surface of airfoil : top
Dyl= double(Yl(x)); % height of c


% because if we start from 0, division will be 0/0, thus NAN,
 eliminate
% that


Dyu(isnan(Dyu)) = 0;

Dyl(isnan(Dyl)) = 0;

Cn = (1/c) * (  TrapezoidalRule(x,Cpl,lower_limit,c,NumPoints,'Cn',0)
 - TrapezoidalRule(x,Cpu,lower_limit,c,NumPoints,'Cn',0)  ) ;

% for Ca
Ca = (1/c) * ( TrapezoidalRule(x,Cpu,lower_limit,c,NumPoints,'Ca',Dyu)
 - TrapezoidalRule(x,Cpl,lower_limit,c,NumPoints,'Ca',Dyl) ) ;


Cl = Cn*cosd(alpha) - Ca*sind(alpha);
Cd = Cn*sind(alpha) + Ca*cosd(alpha);

% compute lift and drag

L = Cl * (1/2) * roh_inf * (V_inf)^2 * c ;
```

```
D = Cd * (1/2) * roh_inf * (V_inf)^2 * c ;

Cl_Conv = Cl; % convergent values
Cd_Conv = Cd; % convergent values
L_conv = Cl_Conv * (1/2) * roh_inf * (V_inf)^2 * c ;
%% change number of points:


j = 1;

for i = 1:1000
    %i;
    x = linspace(lower_limit,c,i); % create segments that we will
 integrate along


Cpu = fnval(Cp_upper, x./c); % evaluate the upper surface Cp's along
 that segment
Cpl = fnval(Cp_lower, x./c); % evaluate the lower surface Cp's along
 that segment

Dyu= double(Yu(x));% evaluate surface of airfoil : top
Dyl= double(Yl(x)); % height of c

% because if we start from 0, division will be 0/0, thus NAN,
 eliminate
% that


Dyu(isnan(Dyu)) = 0;

Dyl(isnan(Dyl)) = 0;



Cn_N(j) = (1/c) * (  TrapezoidalRule(x,Cpl,lower_limit,c,i,'Cn',0) -
 TrapezoidalRule(x,Cpu,lower_limit,c,i,'Cn',0)  ) ;

% for Ca
Ca_N(j) = (1/c) * ( TrapezoidalRule(x,Cpu,lower_limit,c,i,'Ca',Dyu) -
 TrapezoidalRule(x,Cpl,lower_limit,c,i,'Ca',Dyl) ) ;


Cl_N(j) = Cn_N(j)*cosd(alpha) - Ca_N(j)*sind(alpha);
Cd_N(j) = Cn_N(j)*sind(alpha) + Ca_N(j)*cosd(alpha);

% compute lift and drag

L_N(j) = Cl_N(j) * (1/2) * roh_inf * (V_inf)^2 * c ;
D_N(j) = Cd_N(j) * (1/2) * roh_inf * (V_inf)^2 * c ;
Points(j) = i;
  j = j+1;
end
```

```
Panels = Points + 2;
Points = Points .* 2;
%% printout

Tolerance1 = 5/100;
Tolerance2 = 1/100;
Tolerance3 = (1/10)/100;

err_Cl_N = (  abs(Cl_N(1:end) - Cl_Conv ) ./ Cl_Conv ) ;
err_L_N = (  abs(L_N(1:end) - L_conv ) ./ L_conv ) ;

fprintf('-=-=-=-=-=-=( Question 2 ) -=-=-=-=-=-=-=-=-=-=-=-=-');
fprintf('\n \n')

toc

% Printout results
fprintf(['Number of integration points n for L to be within
 ' num2str(Tolerance1.*100) '%%  relative error is: '
 num2str(Points(find(err_Cl_N<Tolerance1,1)))]);
fprintf('\n')

fprintf(['Number of integration points n for L to be within
 ' num2str(Tolerance2.*100) '%%  relative error is: '
 num2str(Points(find(err_Cl_N<Tolerance2,1)))]);
fprintf('\n')

fprintf(['Number of integration points n for L to be within
 ' num2str(Tolerance3.*100) '%%  relative error is: '
 num2str(Points(find(err_Cl_N<Tolerance3,1)))]);
fprintf('\n')


fprintf('\n \n')

fprintf('-=-=-=-=-=-=( END ) -=-=-=-=-=-=-=-=-=-=-=-=-');
fprintf('\n \n \n')



%% plots:
figure(2)
hax=axes;
plot(Points,L_N,'-*k','LineWidth',1);
hold on
line([Points(find(err_L_N<Tolerance1,1))
 Points(find(err_L_N<Tolerance1,1))],get(hax,'YLim'),'Color','r','LineWidth',3)
line([Points(find(err_L_N<Tolerance2,1))
 Points(find(err_L_N<Tolerance2,1))],get(hax,'YLim'),'Color','b','LineWidth',3)
line([Points(find(err_L_N<Tolerance3,1))
 Points(find(err_L_N<Tolerance3,1))],get(hax,'YLim'),'Color','g','LineWidth',3)
x_lim = xlim; % current y-axis limits
plot([x_lim(1) x_lim(2)],[L_conv L_conv],'--c','LineWidth',2)
```

```
legend('Lift','5% error','1% error','0.1% error','Estimated
 convergance Lift','Location','SouthEast')
grid minor
xlabel('Number of points')
ylabel('Lift force [N]')
title('Number of points for Numerical integration Vs. Lift force for
 NACA 0012')

% Up = fnval(Cp_upper, [0:0.001:1]);
% Down = fnval(Cp_lower, [0:0.001:1]);
%
% plot([0:0.001:1],-Up,'LineWidth',2)
% hold on
% plot([0:0.001:1],-Down,'LineWidth',2)
% ylabel('-C_p');
% xlabel('x/c');
% grid minor
```

functions used

```
function [ int ] = TrapezoidalRule(x,y,a,b,N,mode,Dy)
% This function will perform numerical line integration using
% Trapezoidal rule, the user will pass in the interval of integration
% and also the number of segments (panels). the code
% will return the result of the definite integral.
%
% - - - - - - - - - - - - - - - - - - - - - - - - - -
%
%       Inputs:
%               1- x: x interval you integrating alogn
%               2- y: corresponding y results to that x interval
%               3- a: Lower bound of integration
%               4- b: Upper bound of integration
%               5- N: number of segments (panels)
%               6- mode: this mode is specifcally designed for this
 problem,
%                         based on if we integrating for Cn or Ca
%               7- Dy =  y height of airfoil evaluated at each point,
 used
%               only when mode == Ca
%
% - - - - - - - - - - - - - - - - - - - - - - - - - -
%
%       Outputs:
%               1- int: result of definite integral
% - - - - - - - - - - - - - - - - - - - - - - - - - -
%
%
%
%   -Abdulla AlAmeri
```

```
%   -CU Boulder, Fall 2019, ASEN 3111.


% this's just to test numerical integration with symbolic function
% y = subs(f, x);
% ommit if not used


% apply integration
int = 0;
v = 1; % counting


% if we integrating for coefficient of normal force

if mode == "Cn" || mode == "cn"

for k = 1:(N-1)

    series = ( x(k+1) - x(k) ) * (y(k+1) + y(k))/2 ;

    int(v) = series; % store results.

    v = v+1;
end


elseif mode == "Ca" || mode == "ca"

for k = 1:(N-1)

    series = (( x(k+1) - x(k) ) * (y(k+1) + y(k))/2) * (-( Dy(k) -
 Dy(k+1) ) / ( x(k+1) - x(k) )) ;

    int(v) = series; % store results.

    v = v+1;

end


end


int = sum(int);




end
```

Published with MATLAB® R2018a