ASEN 3111 - Computational Assignment 03 - Main

Table of Contents

Produces the lift coefficients and pressure distributions about four NACA airfoils and compares them to thin airfoils

Author: Samuel Razumovskiy
Collaborators: A.Elsayed
Date: 11/8/2019 (last revised: 11/21/2019)

(Knock knock) House keeping

```
clear,clc,close all
```

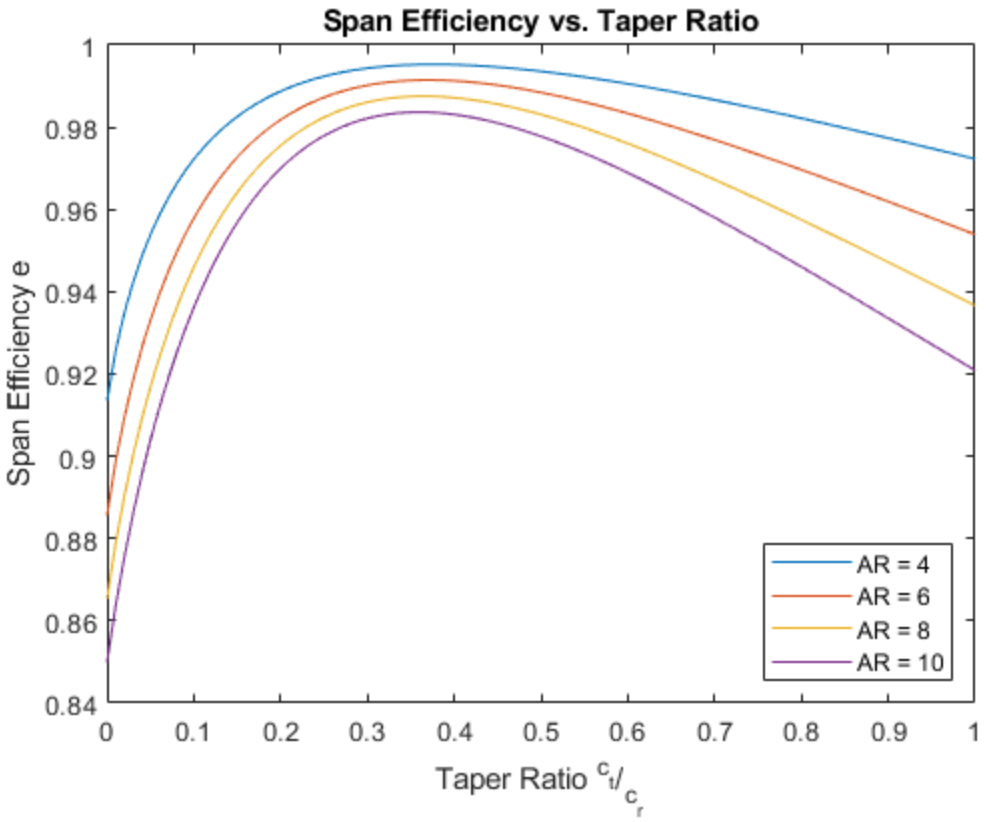
Problem 2

```
% Given constant values
b = 100; % ft
c_t = 5; % ft
c_r = 15; % ft
geo_t = 0; % deg
geo_r = 5; % deg
S = (c_r+c_t)/2*b; % ft^2
N = 100;
v_inf = 150*1.46667; % ft/s
[a0_t,aero_t] = Lift_Slope(0,0,12,c_t,v_inf); % 1/deg, deg
[a0_r,aero_r] = Lift_Slope(2,4,12,c_r,v_inf); % 1/deg, deg


[~,c_L_actual,c_Di_actual] =
 PLLT(b,a0_t,a0_r,c_t,c_r,aero_t,aero_r,geo_t,geo_r,N);
L_actual = 1/2*0.002377*v_inf^2*c_L_actual*S; % lbf
Di_actual = 1/2*0.002377*v_inf^2*c_Di_actual*S; % lbf

% Buncha flags
Dfound5 = false;
Dfound1 = false;
Dfound10 = false;
Cfound5 = false;
Cfound1 = false;
Cfound10 = false;
```

```
% Using the previous absolute value as N = 100 the code find the
 number of
% odd terms needed to get the relative error for lift and drag
for N = 2:100
    [~,c_L,c_Di] =
 PLLT(b,a0_t,a0_r,c_t,c_r,aero_t,aero_r,geo_t,geo_r,N);

    L = 1/2*0.002377*v_inf^2*c_L*S; % lbf
    Di = 1/2*0.002377*v_inf^2c_Di*S; % lbf
    if abs(Di-Di_actual)/Di_actual < 0.05 && Dfound5 == false
        Derror5 = N;
        Dfound5 = true;
    elseif abs(Di-Di_actual)/Di_actual< 0.01 && Dfound1 == false
        Derror1 = N;
        Dfound1 = true;
    elseif abs(Di-Di_actual)/Di_actual< 0.001 && Dfound10 == false
        Derror10 = N;
        Dfound10 = true;
    end
    if abs(L-L_actual)/L_actual < 0.05 && Cfound5 == false
        Cerror5 = N;
        Cfound5 = true;
    elseif abs(L-L_actual)/L_actual< 0.01 && Cfound1 == false
        Cerror1 = N;
        Cfound1 = true;
    elseif abs(L-L_actual)/L_actual< 0.001 && Cfound10 == false
        Cerror10 = N;
        Cfound10 = true;
    end
    if Cfound10 == true && Dfound10 == true
        break
    end
end
fprintf('Lift = %.0f lbf\n',L_actual)
fprintf('Induced Drag = %.0f lbf\n\n',Di_actual)
fprintf('For Lift relative error of 5%% N = %.0f\n',Cerror5)
fprintf('For Lift relative error of 1%% N = %.0f\n',Cerror1)
fprintf('For Lift relative error of 0.1%% N = %.0f\n',Cerror10)
fprintf('For Drag relative error of 5%% N = %.0f\n',Derror5)
fprintf('For Drag relative error of 1%% N = %.0f\n',Derror1)
fprintf('For Drag relative error of 0.1%% N = %.0f\n',Derror10)

Lift = 23687 lbf
Induced Drag = 443 lbf

For Lift relative error of 5% N = 4
For Lift relative error of 1% N = 9
For Lift relative error of 0.1% N = 26
For Drag relative error of 5% N = 7
For Drag relative error of 1% N = 16
For Drag relative error of 0.1% N = 44
```

Problem 3

```
clear,clc,close all

% Vector of tip chord
c_t = linspace(0.001,15)'; % ft
c_r = 15; % ft
% Given aspect ratios
AR = [4,6,8,10];
% Span for a given aspect ratio and tip chord
b = ((AR.*(c_r+c_t))./2); % ft
N = 50;
v_inf = 150*1.46667; % ft/s
a0_r = 2*pi^2/180; % 1/deg
a0_t = 2*pi^2/180; % 1/deg
aero_t = 0; % deg
aero_r = 0; % deg
geo_t = 1; % deg
geo_r = 1; % deg

e = zeros(numel(c_t),numel(AR));
for i = 1:numel(AR)
    for j = 1:numel(c_t)
        [e(j,i),~,~] =
 PLLT(b(j,i),a0_t,a0_r,c_t(j),c_r,aero_t,aero_r,geo_t,geo_r,N);
    end
end
ct_cr = c_t./c_r;
plot(ct_cr,e)
title('Span Efficiency vs. Taper Ratio')
xlabel('Taper Ratio ^{c_t}/_{c_r}')
ylabel('Span Efficiency e')
legend('AR = 4','AR = 6','AR = 8','AR = 10','location','southeast')
```



Functions Called

The following functions were built and called as part of this assignment.

```
function [a,alphaL0] = Lift_Slope(m,p,t,c,v_inf)

N = 150;
[x,y] = NACA_Airfoils(m,p,t,c,N);
alpha = linspace(-5,10);
cl = zeros(size(alpha));
M = N;
for i = 1:100
    cl(i) = Vortex_Panel(alpha(i)*pi/180,v_inf,c,x,y,M);
end
fun = polyfit(alpha,cl,1);
a = fun(1);
alphaL0 = (-fun(2)/fun(1));
end
```

```
function [x,y,yc] = NACA_Airfoils(m,p,t,c,N)
% NACA_Airfoils produces the x and y coordinates of a desired NACA
 four
% digit airfoil and the mean camber line, provided the four digits
 that
% come from m, p, and t, the chord length, and the number of points N
```

```
% Actual values of m, p, and t
m = m/100;
p = p/10;
t = t/100;
% Using the method from Kuthe and Chow to get x values
x = c/2+c/2*cos(linspace(0,pi,ceil((N+1)/2)));

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

% Y mean camber (This somehow works but I'm not sure if it's more
 efficient
% than a regular for loop)

if (0 <= x) & (x <= p*c)
    yc = (m/p^2).*x.*(2*p - x./c);
else
    yc = (m/(1-p)^2).*(c - x).*(1 + x./c - 2*p);
end

% Calculating xi
xi =atan(diff(yc));
xi(end+1) = 0;

% Finding X upper and X lower
xU = x-yt.*sin(xi);
xL = x+yt.*sin(xi);

% Finding Y upper and Y lower then concatenating everything
yU = yc+yt.*cos(xi);
yL = yc-yt.*cos(xi);
x = [xL,fliplr(xU(1:end-1))];
y = [yL,fliplr(yU(1:end-1))];
end

function [e,c_L,c_Di] =
 PLLT(b,a0_t,a0_r,c_t,c_r,aero_t,aero_r,geo_t,geo_r,N)
% PLLT uses prandtl lifting line theorem to caclulate the span
 efficiency,
% the coefficient of lift and the coefficient of drag. Inputs are the
 span,
% lift curve slope at the root and tip, the chord at the root and tip,
 the
% AoA when lift = 0 at the root and tip, the geometric AoA at the root
 and
% tip, and finally the number of odd terms to use in the prandtl
 lifting
% line theorem equation

%% Problem 1
```

```
% A linspace of theta values
theta = linspace(pi/2,pi/(N*2),N)';

% Creating vectors of the lift curve slope, chord, alpha L = 0, and
% geometric alpha for a given theta value
a0 = (a0_r+(a0_t-a0_r)*cos(theta))*180/pi;
c = c_r+(c_t-c_r)*cos(theta);
alphaL0 = (aero_r+(aero_t-aero_r)*cos(theta))*pi/180;
alpha = (geo_r+(geo_t-geo_r)*cos(theta))*pi/180;

% Creating the A matrix and the B vector
B = alpha-alphaL0;
A = zeros(N);
n = (1:2:2*N)';
for i = 1:N
    for j = 1:N
        % PLLT
        p1 = 4*b*(sin(n(j)*theta(i)))/(a0(i)*c(i));
        p2= n(j)*sin(n(j)*theta(i))/sin(theta(i));
        A(i,j) = p1+p2;
    end
end
% Finding the A coefficients
x = A\B;

% Finding the value of delta
delta = sum(n(2:end).*(x(2:end)./x(1)).^2);

% Finding the span efficiency
e = 1/(1+delta);

% Aspect ratio
AR = b/((c_r+c_t)/2);

% Coefficient of lift and drag
c_L = x(1)*pi*AR;
c_Di = c_L^2/(pi*e*AR);
end

function [cl] = Vortex_Panel(alpha,v_inf,c,x,y,M)
% Vortex_Panel uses vortex method to calculate the coefficient of lift
% around a given airfoil provided the AoA, V infinity, chord length, x
 and
% y locations of the airfoil, total number of points M, and a boolean
 value
% for whether or not to plot

MP1    = M+1;
xc     = zeros(1,M);
yc     = zeros(1,M);
S      = zeros(1,M);
theta  = zeros(1,M);
sine   = zeros(1,M);
cosine = zeros(1,M);
```

```
RHS    = zeros(1,M);
Cp     = zeros(1,M);
V      = zeros(1,M);
CN1    = zeros(M,M);
CN2    = zeros(M,M);
CT1    = zeros(M,M);
CT2    = zeros(M,M);
AN     = zeros(M,M);
AT     = zeros(M,M);

for i = 1:M
    ip1       = i+1;
    xc(i)     = 0.5*(x(i)+x(ip1));
    yc(i)     = 0.5*(y(i)+y(ip1));
    S(i)      = sqrt((x(ip1)-x(i))^2 + (y(ip1)-y(i))^2);
    theta(i)  = atan2((y(ip1)-y(i)),(x(ip1)-x(i)));
    sine(i)   = sin(theta(i));
    cosine(i) = cos(theta(i));
    RHS(i)    = sin(theta(i)-alpha);
end

for i = 1:M
    for j = 1:M
        if i == j
            CN1(i,j) = -1;
            CN2(i,j) = 1;
            CT1(i,j) = 0.5*pi;
            CT2(i,j) = 0.5*pi;

        else
            A = -(xc(i)-x(j))*cosine(j)-(yc(i)-y(j))*sine(j);
            B = (xc(i)-x(j))^2+(yc(i)-y(j))^2;
            C = sin(theta(i)-theta(j));
            D = cos(theta(i)-theta(j));
            E = (xc(i)-x(j))*sine(j)-(yc(i)-y(j))*cosine(j);
            F = log(1+S(j)*(S(j)+2*A)/B);
            G = atan2(E*S(j),B+A*S(j));
            P = (xc(i)-x(j))*sin(theta(i)-2*theta(j))+...
                (yc(i)-y(j))*cos(theta(i)-2*theta(j));
            Q = (xc(i)-x(j))*cos(theta(i)-2*theta(j))-...
                (yc(i)-y(j))*sin(theta(i)-2*theta(j));
            CN2(i,j) = D+.5*Q*F/S(j)-(A*C+D*E)*G/S(j);
            CN1(i,j) = .5*D*F+C*G-CN2(i,j);
            CT2(i,j) = C+.5*P*F/S(j)+(A*D-C*E)*G/S(j);
            CT1(i,j) = .5*C*F-D*G-CT2(i,j);
        end
    end
end

for i=1:M
    AN(i,1) = CN1(i,1);
    AN(i,MP1) = CN2(i,M);
    AT(i,1) = CT1(i,1);
    AT(i,MP1) = CT2(i,M);
```

```
    for j=2:M
        AN(i,j) = CN1(i,j)+CN2(i,j-1);
        AT(i,j) = CT1(i,j)+CT2(i,j-1);
    end
end
AN(MP1,1)   = 1;
AN(MP1,MP1) = 1;
for j = 2:M
    AN(MP1,j) = 0;
end
RHS(MP1) = 0;
gamma = AN\RHS';
for i=1:M
    V(i) = cos(theta(i)-alpha);
    for j=1:MP1
        V(i) = V(i)+AT(i,j)*gamma(j);
    end
    Cp(i) = 1-V(i)^2;
end

% Calclating Gamma
rho = 1.225;
pinf = 101325;
Qinf = .5 * rho * v_inf^2;
S = [S,S(1)];
Gamma = sum(2*pi*v_inf*(gamma'.*S));
cl = 2*Gamma/(v_inf*c);
end
```

Published with MATLAB® R2019b