Homework 4

To:	Andrew Ning
From:	Joshua Canlas
Date:	March 23, 2020
Subject:	Homework 4 Memo

Introduction

The purpose of this document is to give a summary report of Homework 4. The first problem is simply reworking the problems that were given in the midterm. The second problem asks us to implement a vortex lattice method. The methods and results for the vortex lattice method are explained in this report.

4.1 Midterm Rework

This problem was turned in a separate submission on Learning Suite.

4.2 Vortex Lattice Method

Introduction

The problem asks us to implement, verify, and validate a vortex lattice method. Vortex lattice method (VLM) is a simple method for calculating the aerodynamic coefficients of a wing. The problem will be split into three parts: (a) verify our numbers for downwash, circulation, and lift coefficient with Bertin's excerpt, (b) verify our induced drag calculation, and (c) validate our VLM against the data in NASA TM-1442. All of the code used to calculate these values is found in Appendix A.

4.2 (a)

Methods

In order to calculate the downwash, we solved for the aerodynamic influence coefficient matrix,

$$AIC_{ij} = \hat{V}_{ij} \cdot \hat{n}_i$$

where $\hat{V}_{ij}$ is the induced velocity at the control point and $\hat{n}_{ij}$ is the normal vector. The normal vector $\hat{n}_{ij}$ can be found using the following equation

$$\hat{n} = \begin{bmatrix} \sin\theta \\ -\cos\theta\sin\phi \\ \cos\theta\cos\phi \end{bmatrix}$$

where θ is the angle of twist and ϕ is the angle of the dihedral. For this problem, we are assuming that there is no angle of twist and that the wing is flat. The induced velocity can be found by using the following equations

$$\hat{V}_{ij} = \frac{1}{4\pi}\left[\frac{\vec{r}_1 \times \vec{r}_2}{(|\vec{r}_1||\vec{r}_2| + \vec{r}_1 \cdot \vec{r}_2)}\left(\frac{1}{|\vec{r}_1|} + \frac{1}{|\vec{r}_2|}\right) + \frac{\vec{r}_1 \times \hat{x}}{(|\vec{r}_1| - r_{1x})}\frac{1}{|\vec{r}_1|} - \frac{\vec{r}_2 \times \hat{x}}{(|\vec{r}_2| - r_{2x})}\frac{1}{|\vec{r}_2|}\right]$$

where $\vec{r_1}$ and $\vec{r_2}$ are defined as

$$\vec{r}_1 = \vec{r}_{CPi} - \vec{r}_j$$
$$\vec{r}_2 = \vec{r}_{CPi} - \vec{r}_{j+1}$$

Each vector points from the corner of the horseshoe vortex at position j to the control point in position i. When the induced velocity vector and normal vector are dotted with each other, we only get the downwash in this case because there is no twist or dihedral.

Once the aerodynamic influence coefficient matrix was solved, we were able to find the circulation distribution and the lift distribution through the following process

$$b_i = -\vec{V}_{ext,i} \cdot \hat{n}_i$$

$$[AIC]\Gamma = b$$

$$L = \rho V_\infty \sum_i \Gamma_i (y_{i+1} - y_i)$$

We were then able to solve for the coefficient of lift once we had the lift distribution.

Results
We only used four panels in this problem so that we could compare our results with Bertin's. After solving for the downwash, circulation and lift distribution, and lift coefficient using this method, we found that our results were the same as Bertin's. While Bertin used a narrower approach, we used a more general approach. The equations that we used can handle different types of wing geometries as well as twist and dihedral. We will use this method again in part (c) when we validate our data against known empirical values of a certain wing.

4.2 (b)

Methods
We know that there are known solutions for specific types of lift distributions. One example is an elliptic lift distribution. We can find it by solving the circulation distribution first, then using the Kutta-Joukowski Theorem, we can find the lift distribution. An elliptic circulation distribution is defined as

$$\Gamma(y) = \Gamma_0 \sqrt{1 - \left(\frac{2y}{b}\right)^2}$$

where Γ_0 is any arbitrary number, y is a control point, and b is the wing span. We then use the Kutta-Joukowski Theorem to solve for the lift distribution

$$L = \rho V_\infty \sum_i \Gamma_i (y_{i+1} - y_i)$$

We also know that the inviscid span efficiency of an elliptic lift distribution is $e_{inv} = 1$. We can solve for the inviscid span efficiency using the following equation

$$e_{inv} = \frac{L^2}{q\pi b^2 D_i}$$

where q is the dynamic pressure and D_i is the induced drag. We can verify our induced drag by plugging it into the inviscid span efficiency equation. If $e_{inv} \approx 1$, then we know that our induced drag is correct. To solve for the induced drag, we used the following equations

$$D_i = \frac{\rho}{2\pi} \sum_{i=1}^{n} \sum_{j=1}^{n+1} \Gamma_i \gamma_j (k_{i,j} - k_{i,-j})$$

where

$$k_{i,\pm j} = \frac{(\pm y_j - \bar{y}_i)(y_{i+1} - y_i) + (z_j - \bar{z}_i)(z_{i+1} - z_i)}{(\pm y_j - \bar{y}_i)^2 + (z_j - \bar{z}_i)^2}$$

and

$$\gamma_i = \begin{cases} -\Gamma_1 & \text{for } i = 1 \\ \Gamma_{i-1} - \Gamma_i & \text{for } i = 2 \ldots n \\ \Gamma_n & \text{for } i = n+1 \end{cases}$$

Results

Fig. 1 shows a convergence plot of e_{inv}. We see that as we increase the number of panels, the inviscid span efficiency converges to 1. This shows that our induced drag calculations were correct. We will use this method for part (c) as well when we validate our results to known values of a wing.

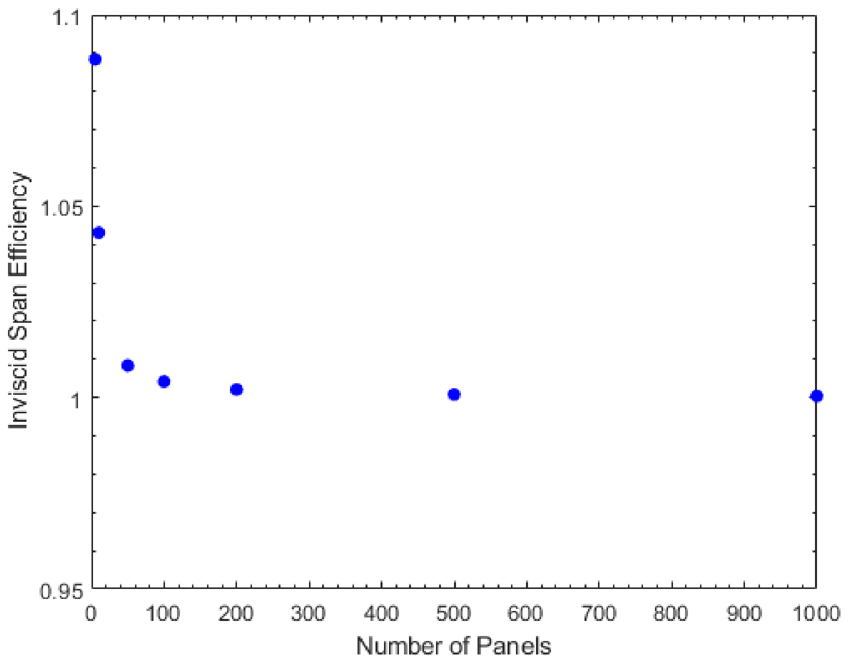


Figure 1: Convergence plot of the inviscid span efficiency. The number of panels used were: 5, 10, 50, 100, 200, 500, and 1000.

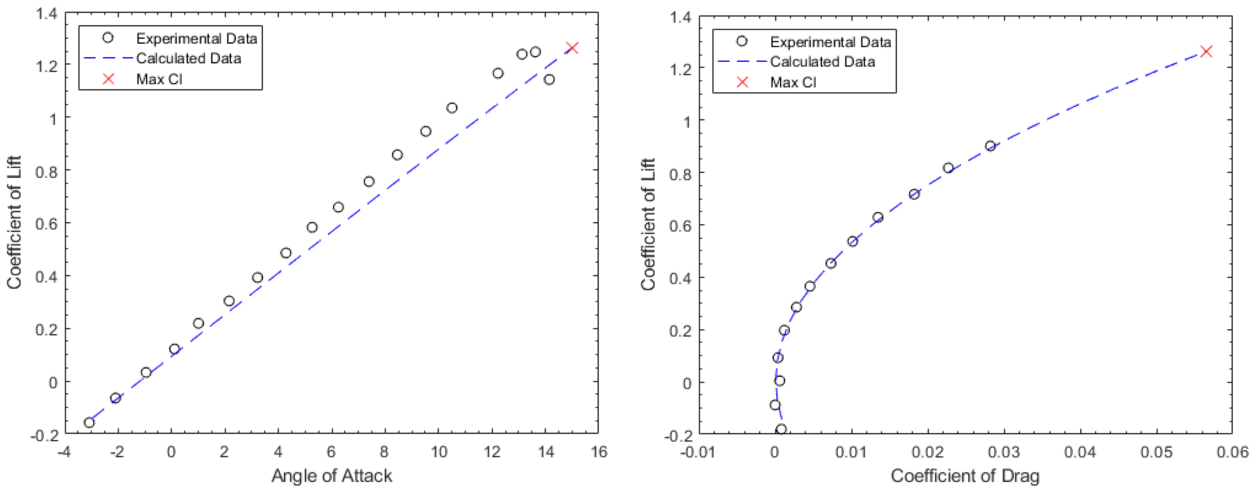
4.2 (c)

Methods

After verifying that both parts (a) and (b) worked, we combined the two to validate our results for the NACA 65-210 sections with 2 degrees washout against known empirical values found in a NASA report. We first used the VLM that we constructed from part (a) to solve for the lift and circulation distribution and coefficient of lift. We then used the circulation distribution that was solved from the VLM to find the induced drag and drag coefficient. These were done using various angles of attack.

Results

Figs. 2a and 2b show plots of angle of attack versus coefficient of lift and coefficient of drag versus coefficient of lift. They also show where the maximum coefficient of lift occurs. Notice that in Fig. 2a the empirical data shows that at about 13 degrees, it begins to stall, whereas our computed values show that the coefficient of lift continues to increase. This is because we did not account for stall in our calculations.



(a) *(b)*

Figure 2: (a) Plot of angle of attack vs. coefficient of lift. (b) Plot of coefficient of drag vs coefficient of lift. The experimental data that are used to compare in these plots were taken from the NASA report from the homework problem.

Overall, as can be seen in these plots, the calculated and experimental data match fairly well. In Fig. 2a, the calculated data deviates from the experimental data a little bit, but not by very much. It is actually pretty surprising to see how close these data match with each other, especially when we didn’t even take into account the 3 degree dihedral that the wing had. Therefore, we can conclude that the dihedral didn’t really affect the aerodynamic properties of this wing.

Appendix A

VLM Code:

```
clc
clear all

points = 20;
alpha = linspace(-3,15,points);

for X = 1:points
    N = 100; % number of panels
    b = 180; % 1 % inches; wing span
    alpha_curr = alpha(X); % make into an array of varying angles
    beta = 0;
    phi = zeros(1,N); % dihedral angle
    theta = linspace(2,0,N); % [0, 0, 0]; %angle of twist
    sweep = 30; % 45
    max_span = 0.5 * b;
    lat_spacing = max_span / N;
    center_spacing = lat_spacing / 2;
    rho = 1.225;
    c_cp = linspace(28.57, 11.43, N); % linspace((0.2*b),(0.2*b),N);
    c_bv = linspace(28.57, 11.43, N+1); % linspace((0.2*b), (0.2*b), N+1));
    c_avg = (28.57+11.43)/2; % (0.2 * b);
    S = b*c_avg;
    x_hat = [1; 0; 0];

    V_inf = [cosd(alpha_curr)*cosd(beta);
        -sind(beta);
        sind(alpha_curr)*cosd(beta)];

    n_hat = [sind(theta); ...
        -cosd(theta).*sind(phi); ...
        cosd(theta).*cosd(phi)]; % normal vector

    % control points and vortex points
    % s = starboard; p = port
    y_s(1) = 0;
    x(1) = 0;
    for R = 1:N
        if R == 1
            ybar_s(R) = center_spacing;
        else
            ybar_s(R) = ybar_s(R-1) + lat_spacing;
        end
        y_s(R+1) = y_s(R) + lat_spacing;
    end

    xbar = (tand(sweep)*ybar_s)+((3/4)*c_cp);
    x = (tand(sweep)*y_s)+((1/4)*c_bv);
    y_p = -y_s;

    r_s = [x; y_s; zeros(1,N+1)];
    r_p = [x; y_p; zeros(1,N+1)];
```

```
rcp = [xbar; ybar_s; zeros(1,N)];

for I = 1:N
    for J = 1:N
        r1 = rcp(:,I) - r_s(:,J);
        r2 = rcp(:,I) - r_s(:,J+1);
        const = 1 / (4*pi);
        temp1 = cross(r1, r2) / (norm(r1)*norm(r2) + dot(r1, r2));
        temp2 = (1/norm(r1)) + (1/norm(r2));
        temp3 = cross(r1,x_hat)/((norm(r1)-r1(1)))*(1/norm(r1));
        temp4 = cross(r2,x_hat)/((norm(r2)-r2(1)))*(1/norm(r2));
        V_hat = const*(temp1*temp2 + temp3 - temp4);
        AIC_s = dot(V_hat, n_hat(:,I));

        r1 = rcp(:,I) - r_p(:,J);
        r2 = rcp(:,I) - r_p(:,J+1);
        temp1 = cross(r1, r2) / (norm(r1)*norm(r2) + dot(r1, r2));
        temp2 = (1/norm(r1)) + (1/norm(r2));
        temp3 = cross(r1,x_hat)/((norm(r1)-r1(1)))*(1/norm(r1));
        temp4 = cross(r2,x_hat)/((norm(r2)-r2(1)))*(1/norm(r2));
        V_hat = const*(temp1*temp2 + temp3 - temp4);
        AIC_p = dot(V_hat, n_hat(:,I));

        AIC(I,J) = AIC_s - AIC_p;
    end
    B(I) = dot(-V_inf,n_hat(:,I));
end

gamma = AIC \ B';

% Kutta-Joukowski Theorem
L = 0;
for R = 1:N
    L = L + gamma(R)  * (y_s(R+1) - y_s(R));
end

L = 2 * L * 1 * rho;
cL(X) = L / (0.5 * rho * 1^2 * S);

% % make elliptical lift distribution
% L_ellip = 0;
% for R = 1:N
%     gamma_ellip(R) = sqrt(1 - ((2*ybar_s(R))/b)^2);
%     L_ellip = L_ellip + gamma_ellip(R) * (y_s(R+1) - y_s(R));
% end
%
% L_ellip = 2 * L_ellip * rho * 1;

% solve for kij
D = 0;
temp = 0;
for I = 1:N
    for J = 1:N+1
        kij_s = ((y_s(J)-ybar_s(I)) * (y_s(I+1)-y_s(I))) / ...
            (y_s(J) - ybar_s(I))^2;
            kij_p = ((y_p(J)-ybar_s(I)) * (y_p(I+1)-y_p(I))) / ...
                (y_p(J) - ybar_s(I))^2;
            if J == 1
                small_gamma = -gamma(1);
            elseif J == N+1
                small_gamma = gamma(N);
            else
                small_gamma = gamma(J-1) - gamma(J);
            end
            D = D + gamma(I) * small_gamma * (kij_s + kij_p);
        end
    end
    D = (rho / (2*pi)) * D;
    cD(X) = D / (0.5 * rho * 1^2 * S);
    q = 0.5 * rho * 1;
    e_inv = L^2 / (q * pi * b^2 * D);
end

CD_cL = [0.007482262815025453, -0.18027073690930484;
    0.0061416363187905, -0.08790679453883343;
    0.005630921463081973, 0.004422326364203588;
    0.004709603470539586, 0.09257287117507929;
    0.005015742205069423, 0.197455131088315 7;
    0.007414070774632547, 0.28546639002945273;
    0.01065101635158075, 0.36505085385139924;
    0.013879256561670245, 0.4530272913251019;
    0.01752680527545228, 0.5367903312392088;
    0.021580604442638873, 0.6289279340713548;
    0.02688362375404435, 0.7168173178764708;
    0.032588540835425034, 0.8172772514255037;
    0.039555736111312606, 0.9009010054698721];
CD0_cL = [0.006661022468251384, -0.16507109519157703
    0.006134809508303485, -0.08118962335829805;
    0.005019863236730706 5, 0.019971779007923463;
    0.004326929338977531, 0.10402691848474976;
    0.003799413871703028, 0.19207641376316054;
    0.004604363399544121, 0.28290459133832635;
    0.006078801693259525, 0.36470205144903933;
    0.006551611852816673, 0.4517095408661673;
    0.00735786388798437, 0.5383696949962009;
    0.008082057961576034, 0.6209486595028763;
    0.008636926082709214, 0.7120373385433625;
    0.009862585477043311, 0.7899272766742644;
    0.011331813741452302, 0.8883968305655051];

alpha_cL = [-3.0811361981063428, -0.1573780043699926;
    -2.10050983248361, -0.06438455935906795;
    -0.9541150764748778, 0.032920611798980426;
    0.11070648215585521, 0.12177713037144944;
    1.0071376547705668, 0.21890750182083019;
    2.15615440640931 4, 0.30362709395484333;
    3.2209759650400542, 0.39248361252731256;
    4.284923525127454, 0.4855353241077933;
    5.264675892206846, 0.5827239621267297;
    6.248798252002906, 0.6589366351056082;
    7.395193008011653, 0.7562418062636563;
    8.457392571012377, 0.8576839038601602;
    9.522214129643118, 0.9465404224326294;
    10.50371449380917, 1.0353386744355428;
    12.22636562272396, 1.1666132556445739;
    13.128040786598689, 1.238572469045885;
    13.626219956300076, 1.2473124544792427;
    14.147997086671523, 1.1427822286962854];

cD_exp = CD_cL(:,1) - CD0_cL(:,1);
figure(1)
plot(cD_exp, CD_cL(:,2), 'ko')
hold on
plot(cD, cL, 'b--')
hold on
plot(cD(end), cL(end), 'rx', 'MarkerSize', 10)
xlabel('Coefficient of Drag')
ylabel('Coefficient of Lift')
set(gca,'XMinorTick','on','YMinorTick','on')
legend('Experimental Data', 'Calculated Data', ...
    'Max Cl', 'Location', 'northwest')
xlim([-0.01, 0.06])

figure(2)
plot(alpha_cL(:,1), alpha_cL(:,2), 'ko')
hold on
plot(alpha, cL, 'b--')
hold on
plot(alpha(end), cL(end), 'rx', 'MarkerSize', 10)
xlabel('Angle of Attack')
ylabel('Coefficient of Lift')
set(gca,'XMinorTick','on','YMinorTick','on')
legend('Experimental Data', 'Calculated Data', ...
    'Max Cl', 'Location', 'northwest')

% e_inv_plot = [1.0884, 1.0431, 1.0084, 1.0042,...
%     1.0021, 1.0008, 1.0004]; % 5,10,50,100,200,500,1000
% figure(1)
% plot([5,10,50,100,200,500,1000], e_inv_plot, 'b.', 'MarkerSize', 20)
% ylim([0.95,1.10])
% xlabel('Number of Panels')
% ylabel('Inviscid Span Efficiency')
% set(gca,'XMinorTick','on','YMinorTick','on')
```