MATHS IN ACTION B – ASSIGNMENT 2

LUKE APPLEBY

Abstract. The affects angle of attack and wing geometry have on the wing performance metrics lift and drag were investigated computationally. It was found that greater positive curvature on the topside of the wing gave rise to increased lift. The relationship between angle of attack and lift was found to be sinusoidal however for angles in the interval $[0, 0.3]$ this relationship was close to positive linear. The data collected on how attack angle or wing geometry affect drag were insufficient to draw conclusions from.

1. Introduction

Understanding the flow of air around the wings of an aircraft is essential for efficient and safe air travel. There are many factors that affect the flow such as angle of attack, velocity of airflow, texture of the wing surface, whether it is a laminar or chaotic flow and the geometry of the wing. In this report we use a computer simulation to investigate how the factors angle of attack and wing geometry affect the lift and drag generated by a wing. Angle of attack is of particular importance since it is known that at low speeds an excessive angle of attack can cause flow separation above the wing which may result in a fatal crash. We aim for the results of this investigation to be informative in aircraft design and flight.

2. Background Information

2.1. **Models.** The flow around a wing in 2-dimensions can be described by the composition of the following two functions in the complex plane. Firstly:

$$\hat{\omega}(\zeta) = -(U - iV)(\zeta - \zeta_0) - (U + iV)\frac{c^2}{(\zeta - \zeta_0)} - \frac{i\kappa}{2\pi} \cdot \log\frac{\zeta - \zeta_0}{c} \tag{1}$$

models the flow of a fluid around the disk $D_c(\zeta_0)$. We use a second map to deform the disk into a shape that more closely resembles a wing:

$$f(\zeta) = \zeta + \frac{\lambda^2}{\zeta}. \tag{2}$$

Letting $\zeta_0 = -\lambda + ce^{i\beta}$ allows us to vary the position of ζ_0 which in turn corresponds to a different wing geometry. $z = f(\zeta)$ and $\omega(z) = \hat{\omega}(\zeta)$ so we have $\omega(z) = \hat{\omega}(f^{-1}(z))$. To properly model the flow ω must be continuous so it's important to know the potential points of discontinuity. We find f^{-1} by substituting z for $f(\zeta)$ and solving for ζ: $z = \zeta + \frac{\lambda^2}{\zeta} \Rightarrow z\zeta = \zeta^2 + \lambda^2 \Rightarrow \zeta^2 - z\zeta + \lambda^2 = 0$. This is a quadratic in ζ therefore the quadratic formula gives us:

$$\zeta = \frac{z \pm \sqrt{z^2 - 4\lambda^2}}{2} \text{ that is: } f^{-1}(\zeta) = \frac{\zeta \pm \sqrt{\zeta^2 - 4\lambda^2}}{2} \tag{3}$$

Notice when $\zeta = \pm 2\lambda$ there is a zero in the square root. Any function equal to a branch of square root is discontinuous in some line segment originating at 0. This is to say that $f^{-1}(\zeta)$ has singularities at $\pm 2\lambda$. We have $z = f(\zeta)$ and see that $\pm 2\lambda = \zeta + \frac{\lambda^2}{\zeta} \iff 2\lambda\zeta = \zeta^2 + \lambda^2 \iff \zeta^2 - 2\lambda\zeta + \lambda^2 = 0 \iff \zeta = \pm\lambda$.

Consider the first singularity at $\zeta = \lambda$ with $\zeta_0 = -\lambda + ce^{i\beta}$ and $c\cos(\beta) > \lambda$. I claim that $\zeta = \lambda$ is in the disk $D_c(\zeta_0) = \{\zeta : |\zeta - \zeta_0| < c\}$. See that:

$$\begin{aligned}|\zeta - \zeta_0| &= |\lambda - (-\lambda + ce^{i\beta})| \\ &= |2\lambda - ce^{i\beta}| \\ &= |2\lambda - c\cos(\beta) - ic\sin(\beta)| \\ &= \left((2\lambda - c\cos(\beta))^2 + (c\sin(\beta))^2\right)^{\frac{1}{2}}.\end{aligned}$$

Notice that $((c\cos(\beta))^2 + (c\sin(\beta))^2)^{\frac{1}{2}} = c$ and $c\cos(\beta) > \lambda \Rightarrow |2\lambda - c\cos(\beta)| < |c\cos(\beta)|$ therefore $|\zeta - \zeta_0| < c$. That is to say $\zeta = \lambda \in D_c(\zeta_0)$.

Now consider the point $\zeta = -\lambda$. The circle ∂D is the set of points $\zeta \in \mathbb{C}$ such that $\zeta = (-\lambda + ce^{i\beta}) + ce^{i\alpha}$ for $\alpha \in [0, 2\pi)$. Notice that $e^{\pi i} = -1$ therefore:

$$ce^{i\beta} \cdot e^{\pi i} = ce^{i(\pi+\beta)} = -ce^{i\beta}.$$

So for $\alpha = \pi + \beta + 2k\pi$ for some $k \in \mathbb{Z}$ we have $\zeta = (-\lambda + ce^{i\beta}) + ce^{i(\pi+\beta)} = -\lambda$ thus $-\lambda \in \partial D$.

2.2. **Circulation.** We have the velocity field

$$u - iv = \omega'(z) = \hat{\omega}'(\zeta)/f'(\zeta) = \frac{\hat{\omega}'(\zeta)}{1 - \frac{\lambda^2}{\zeta^2}}.$$

So at the trailing edge $\zeta = -\lambda$ we have a zero in the denominator of the expression thus $\omega'(z) := \infty$ unless $\hat{\omega}'(\zeta)$ has a zero of same order at $-\lambda$. So by letting $\hat{\omega}'(\zeta) = 0$ and $\zeta = -\lambda$ we can find κ that will result in finite velocity.

Differentiating $\hat{\omega}$ in terms of zeta gives:

$$\hat{\omega}'(\zeta) = -(U - iV) + (U + iV)\frac{c^2}{(\zeta - \zeta_0)^2} - \frac{i\kappa}{2\pi} \cdot (\zeta - \zeta_0)^{-1}.$$

Notice that $\zeta - \zeta_0 = -\lambda - (-\lambda + ce^{i\beta}) = -ce^{i\beta}$ therefore:

$$\begin{aligned}0 &= -(U - iV) + (U + iV)\frac{c^2}{(-ce^{i\beta})^2} - \frac{i\kappa}{2\pi} \cdot (-ce^{i\beta})^{-1} \\ \Rightarrow \frac{i\kappa}{2\pi} &= \left((U - iV) - (U + iV)\frac{1}{e^{i2\beta}}\right) \cdot ce^{i\beta} \\ &= (U - iV)ce^{i\beta} - (U + iV)ce^{-i\beta}\end{aligned}$$

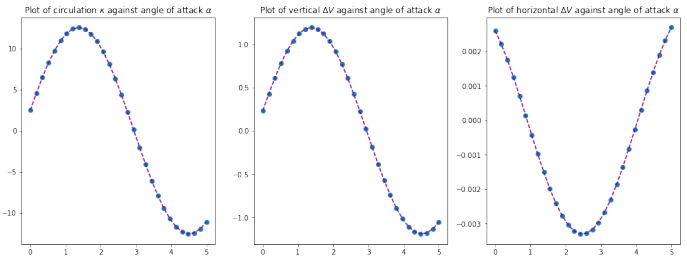
Substituting $U + iV = |U|e^{-i\alpha}$ and $U - iV = |U|\overline{e^{-i\alpha}} = |U|e^{i\alpha}$:

$$\begin{aligned}\Rightarrow \frac{i\kappa}{2\pi} &= c|U|e^{i(\alpha+\beta)} - c|U|e^{-i(\alpha+\beta)} \\ &= c|U|(\cos(\alpha+\beta) + i\sin(\alpha+\beta) - \cos(-\alpha-\beta) - i\sin(-\alpha-\beta)) \\ &= c|U|(2i\sin(\alpha+\beta)) \Rightarrow \kappa = 4\pi c|U|\sin(\alpha+\beta). \qquad (4)\end{aligned}$$

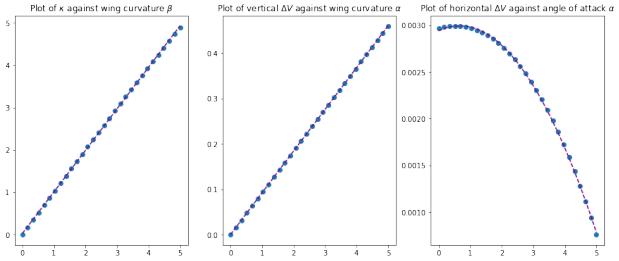
This expression for κ was used when implementing equation 2.1 in our code A.

3. Results and methodology

We looked at the relationship between β, which corresponds to the upward curvature of the wing, and κ the circulation. It has been shown in the second Maths in Action workshop [2] that the lift generated by the wing is $|U|\kappa$. $|U|$ is unchanged in our investigation so we will assume it to be 1 as convention therefore lift is entirely dependant on κ. Additionally we defined a function `delta_v` in A which measures the difference between the mean air velocity before and after the wing in either the



(a) Angle of attack α plotted against lift on the left κ, against ΔV in the middle and against drag on the right.



(b) Wing curvature β plotted against lift on the left κ, against ΔV in the middle and against drag on the right.

Figure 1

vertical or horizontal direction. Motivated by $F = ma = m \cdot \frac{\Delta V}{\Delta t}$ we used ΔV as a second measure for lift and also to measure drag. Using 4 to calculate κ for β evenly spaced in the interval $[0, 0.4]$ we plot the relationship as seen in figure 1b. A linear relationship between wing curvature and lift was observed additionally the horizontal ΔV (drag) decreased quadratically as curvature increased. We also looked at angle of attack α and lift. The same methods as before were used but we varied $\alpha \in [0, 5]$ by even increments. The relationship of angle of attack with κ, vertical ΔV and horizontal ΔV is seen in figure 1a. A sinusoidal relationship was observed between α and both our measures for lift κ and ΔV. The horizontal ΔV is also plotted and its relationship with α fits a cosinusoidal curve.

4. Discussion and conclusions

It was observed that up to a point increasing angle of attack resulted in greater lift produced by the wing and after $\alpha = 1.5$ lift decreases. The angle at which this decrease occurs is far greater than what is observed elsewhere with 0.26 radians being the typical stall angle for most wings [3]. This may be because in our code the wing was held fixed against a constant airflow whereas in aircraft stalling often occurs when there is a loss of velocity in a semi-vertical climb [4]. We found that vertical ΔV up to a re-scaling agreed with κ which suggests it is a suitable measure of lift. In the case of horizontal ΔV it decreased as wing curvature increased which is plausible since negative ΔV means the airflow moving slower after the wing than before indicating a drag force on the wing. Contrary to our expectations horizontal ΔV was positive for all values of β which may lead us to believe there was a negative drag vector however this is not the case. In actuality this is most likely a result of an 'air cushion' building up in front of the wing causing the velocity before the wing to be less than the velocity once the air had cleared the wing. In future investigations velocity of airflow could be measured further before the wing to account for this in ΔV calculations.

References

[1] Maths in Action lecture notes.
[2] Maths in Action workshop.
[3] SKYbrary article: Stall
`https://www.skybrary.aero/articles/stall#:~:text=A%20stall%20occurs%20when%20the,is%20typically%20around%2015%C2%B0.`
[4] Aviation dictionary: Vertical Stall
`https://aviation_dictionary.en-academic.com/7149/vertical_stall`

Appendix A. Codes

A.1. Code for finding ΔV.

```
def delta_v(V):
    # Find the initial mean velocity of the velocity field before the wing
    mean_b = V[:,-5:].mean()

    # Find the velocity of the velocity field after the wing
    mean_a = V[:,:5].mean()

    # Compute mean delta V
    dV = mean_b - mean_a

    # Calculate force using the formula F = m * deltaV/deltaT and assume a mass per unit tim
    # i.e. the speed and denisty of the fluid
    return dV
```

A.2. **Code for plotting various wing shapes (plots not included in report).**

```
# Shape of wing vs lift kappa

import numpy as np
import matplotlib.pyplot as plt

n = 100
theta = np.linspace(0,2*np.pi,n)
c = 1
lams = np.linspace(0.8, 0.95, 4)
betas = np.linspace(0, 0.3, 4 )
betas = np.round(betas, decimals = 2)

Z = []
X = []
Y = []
kappas = []


for i in range(4):
    for j in range(4):
        zeta0 = - lams[i] + c * np.exp(1j*betas[j])
        zeta = zeta0 + c * np.exp(1j*theta)
        z = zeta + lams[i]**2/zeta
        x = np.real(z)
        y = np.imag(z)
        Z.append(z)
        X.append(x)
        Y.append(y)
        alpha = 0
        W = 1 * np.exp(-1j*alpha)
        kappa = 4*np.pi*c*np.abs(W)*np.sin(alpha+betas[j])
        kappas.append(kappa)

kappas = np.round(kappas, decimals = 3)
```

```
fig, ax = plt.subplots(4,4, figsize=(12,12))

for i in range(16):
    ax[i//4,i%4].plot(X[i],Y[i], color='green')
    Bval = str(betas[i%4])
    Lval = str(lams[i//4])
    kap = str(kappas[i])
    ax[i//4,i%4].set_title(r'$\beta = $' + Bval + ', ' r'$\lambda =
    $' + Lval + ' ' +  r'$\kappa = $' + kap)
    ax[i//4,i%4].axis('off')

plt.axis('equal')
plt.setp(ax, ylim=(-2,1), xlim=(-2,2))
plt.tight_layout()
plt.show()
```

A.3. **Code used to plot flow of air around wing.**

```
# OG code also

# Parameters
lam = 0.9
beta = 0.2
alpha = 0
lam = 0.9


W = 1 * np.exp(-1j*alpha)
kappa = 0
kappa = 4*np.pi*c*np.abs(W)*np.sin(alpha+beta)
# circulation for finite velocity at trailing edge

# Grid in zeta-plane
zeta0 = - lam + c * np.exp(1j*beta)
zeta = zeta0 + c * np.exp(1j*theta)
m = 25
zetar = np.linspace(np.real(zeta0)-3*c,np.real(zeta0)+3*c,m)
zetai = np.linspace(np.imag(zeta0)-3*c,np.real(zeta0)+3*c,m)
xi, eta = np.meshgrid(zetar,zetai)
zeta = xi + 1j * eta
w = - np.conjugate(W) * (zeta - zeta0) - c**2 * W / (zeta - zeta0) \
    - 1j*kappa/(2*np.pi)*np.log((zeta-zeta0)/c)

# Streamfunction
psi = np.imag(w)
psi[(xi-np.real(zeta0))**2+(eta-np.imag(zeta0))**2<c**2] = np.nan

# Velocity
dwdzeta = - np.conjugate(W)  + c**2 * W / (zeta - zeta0)**2 \
```

```
    - 1j*kappa/(2*np.pi)/(zeta-zeta0)

# Grid in the z-plane, image of the grid in the zeta-plane
z = zeta + lam**2/zeta
x = np.real(z)
y = np.imag(z)
dzdzeta = 1 - lam**2/zeta**2

# Velocity field
dwdz = dwdzeta/dzdzeta
u = np.real(dwdz)
v = - np.imag(dwdz)
x[(xi-np.real(zeta0))**2+(eta-np.imag(zeta0))**2<c**2] = np.nan
u[(xi-np.real(zeta0))**2+(eta-np.imag(zeta0))**2<c**2] = np.nan
v[(xi-np.real(zeta0))**2+(eta-np.imag(zeta0))**2<c**2] = np.nan

# Plots
plt.quiver(x,y,u,v)
plt.axis('scaled')
plt.xlabel(r'$x$')
plt.ylabel(r'$y$')
# Plot image of circle
zetawing = zeta0 + c * np.exp(1j*theta)
zwing = zetawing + lam**2/zetawing
plt.plot(np.real(zwing),np.imag(zwing))
plt.show()

delta_v(v)
```

A.4. Code for plotting angle of attack α against κ and vertical ΔV.

```
from scipy.optimize import curve_fit

# Parameters
lam = 0.9
alphas = np.linspace(0, 5, 30)
dVs = []
drags = []
kappas = []

for i in range(30):

    beta = 0.2
    alpha = alphas[i]
    W = 1 * np.exp(-1j*alpha)
    kappa = 4*np.pi*c*np.abs(W)*np.sin(alpha+beta)
    # circulation for finite velocity at trailing edge

    # Grid in zeta-plane
```

```
    zeta0 = - lam + c * np.exp(1j*beta)
    zeta = zeta0 + c * np.exp(1j*theta)
    m = 25
    zetar = np.linspace(np.real(zeta0)-3*c,np.real(zeta0)+3*c,m)
    zetai = np.linspace(np.imag(zeta0)-3*c,np.real(zeta0)+3*c,m)
    xi, eta = np.meshgrid(zetar,zetai)
    zeta = xi + 1j * eta
    w = - np.conjugate(W) * (zeta - zeta0) - c**2 * W / (zeta - zeta0) \
        - 1j*kappa/(2*np.pi)*np.log((zeta-zeta0)/c)

    # Streamfunction
    psi = np.imag(w)
    psi[(xi-np.real(zeta0))**2+(eta-np.imag(zeta0))**2<c**2] = np.nan

    # Velocity
    dwdzeta = - np.conjugate(W)  + c**2 * W / (zeta - zeta0)**2 \
        - 1j*kappa/(2*np.pi)/(zeta-zeta0)

    # Grid in the z-plane, image of the grid in the zeta-plane
    z = zeta + lam**2/zeta
    x = np.real(z)
    y = np.imag(z)
    dzdzeta = 1 - lam**2/zeta**2

    # Velocity field
    dwdz = dwdzeta/dzdzeta
    u = np.real(dwdz)
    v = - np.imag(dwdz)
    x[(xi-np.real(zeta0))**2+(eta-np.imag(zeta0))**2<c**2] = np.nan
    u[(xi-np.real(zeta0))**2+(eta-np.imag(zeta0))**2<c**2] = np.nan
    v[(xi-np.real(zeta0))**2+(eta-np.imag(zeta0))**2<c**2] = np.nan

    dVs.append(delta_v(v))
    drags.append(delta_v(u))
    kappas.append(kappa)


def func(x, a, b):
    return a*np.sin(x+b)

fig, ax = plt.subplots(1,3, figsize = (17,6))

popt0, pcov0 = curve_fit(func, alphas, kappas)
popt1, pcov1 = curve_fit(func, alphas, dVs)
popt2, pcov2 = curve_fit(func, alphas, drags)


ax[0].plot(alphas, func(alphas, *popt0), color='purple', linestyle='--',
```

```
label="Fitted Curve")
ax[0].scatter(alphas, kappas)
ax[0].set_title(r'Plot of circulation $\kappa$ against angle of attack $\alpha$')
ax[1].plot(alphas, func(alphas, *popt1), color='purple', linestyle='--',
label="Fitted Curve")
ax[1].scatter(alphas, dVs)
ax[1].set_title(r'Plot of vertical $\Delta V$ against angle of attack $\alpha$')
ax[2].plot(alphas, func(alphas, *popt2), color='purple', linestyle='--',
label="Fitted Curve")
ax[2].scatter(alphas, drags)
ax[2].set_title(r'Plot of horizontal $\Delta V$ against angle of attack $\alpha$')
plt.show()
```

A.5. **Code for plotting wing curvature β against κ and vertical ΔV.**

```
from scipy.optimize import curve_fit

# Parameters
lam = 0.9
betas = np.linspace(0, 0.4, 30)
dVs = []
kappas = []
drags = []

for i in range(30):

    beta = betas[i]
    alpha = 0
    W = 1 * np.exp(-1j*alpha)
    kappa = 4*np.pi*c*np.abs(W)*np.sin(alpha+beta)
    # circulation for finite velocity at trailing edge

    # Grid in zeta-plane
    zeta0 = - lam + c * np.exp(1j*beta)
    zeta = zeta0 + c * np.exp(1j*theta)
    m = 25
    zetar = np.linspace(np.real(zeta0)-3*c,np.real(zeta0)+3*c,m)
    zetai = np.linspace(np.imag(zeta0)-3*c,np.real(zeta0)+3*c,m)
    xi, eta = np.meshgrid(zetar,zetai)
    zeta = xi + 1j * eta
    w = - np.conjugate(W) * (zeta - zeta0) - c**2 * W / (zeta - zeta0) \
        - 1j*kappa/(2*np.pi)*np.log((zeta-zeta0)/c)

    # Streamfunction
    psi = np.imag(w)
    psi[(xi-np.real(zeta0))**2+(eta-np.imag(zeta0))**2<c**2] = np.nan

    # Velocity
    dwdzeta = - np.conjugate(W)  + c**2 * W / (zeta - zeta0)**2 \
```

```
        - 1j*kappa/(2*np.pi)/(zeta-zeta0)

    # Grid in the z-plane, image of the grid in the zeta-plane
    z = zeta + lam**2/zeta
    x = np.real(z)
    y = np.imag(z)
    dzdzeta = 1 - lam**2/zeta**2

    # Velocity field
    dwdz = dwdzeta/dzdzeta
    u = np.real(dwdz)
    v = - np.imag(dwdz)
    x[(xi-np.real(zeta0))**2+(eta-np.imag(zeta0))**2<c**2] = np.nan
    u[(xi-np.real(zeta0))**2+(eta-np.imag(zeta0))**2<c**2] = np.nan
    v[(xi-np.real(zeta0))**2+(eta-np.imag(zeta0))**2<c**2] = np.nan

    dVs.append(delta_v(v))
    kappas.append(kappa)
    drags.append(delta_v(u))


def func(x, a, b):
    return a + x*b

def func2(x, a, b, c):
    return a + x*b + x*x*c

fig, ax = plt.subplots(1,3, figsize = (15,6))

popt0, pcov0 = curve_fit(func, alphas, kappas)
popt1, pcov1 = curve_fit(func, alphas, dVs)
popt2, pcov2 = curve_fit(func2, alphas, drags)


ax[0].plot(alphas, func(alphas, *popt0), color='purple', linestyle='--',
label="Fitted Curve")
ax[0].scatter(alphas, kappas)
ax[0].set_title(r'Plot of $\kappa$ against wing curvature $\beta$')
ax[1].plot(alphas, func(alphas, *popt1), color='purple', linestyle='--',
label="Fitted Curve")
ax[1].scatter(alphas, dVs)
ax[1].set_title(r'Plot of $\Delta V$ against wing curvature $\alpha$')
ax[2].plot(alphas, func2(alphas, *popt2), color='purple', linestyle='--',
label="Fitted Curve")
ax[2].scatter(alphas, drags)
ax[2].set_title(r'Plot of horizontal $\Delta V$ against angle of attack $\alpha$')
plt.show()
```