AE667-Assignment3

Manoj Bhadu 170010036

March 2021

Contents

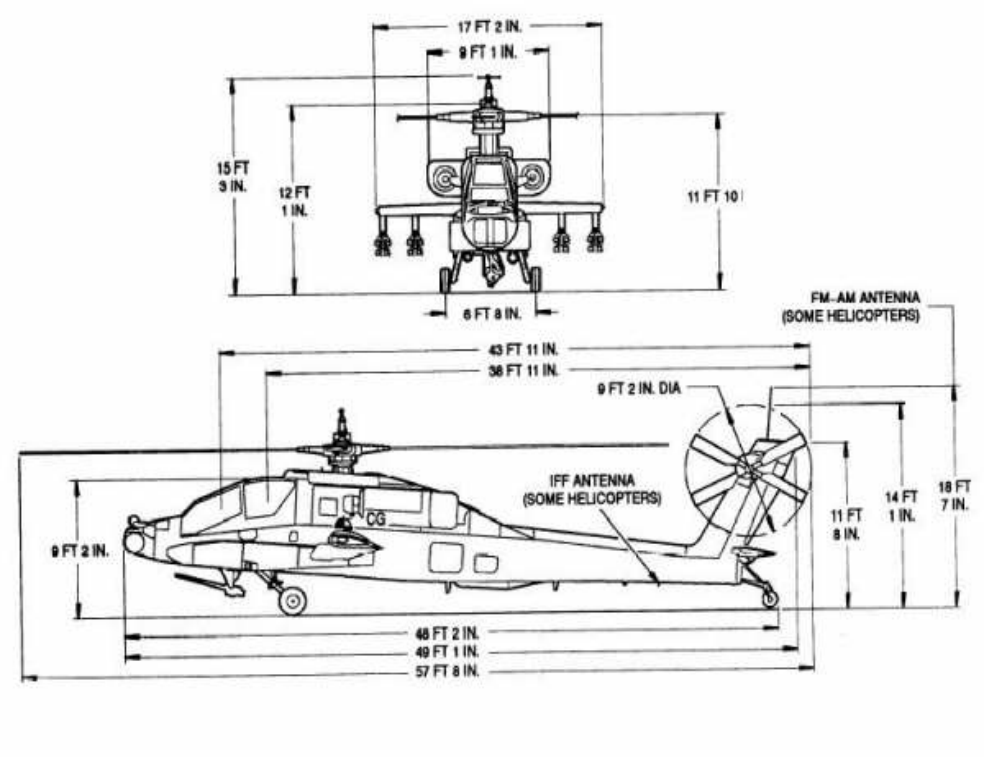
1 Status of program

Program is working fine.

2 Starting Data

2.1 Parameters

Parameters	Value
Density(Kg/m^3)	1.215
Weight(Kg)	8000
R	7.3
Rrc	0.9
RPM	260
Number of blades	4
a	5.8
C_d	0.8
Taper(m)	0.55
Plate Area(m^2)	6.8
Forward Velocity(V)(m/s)	73



2.2 Assumptions

- No Swirl
- compressibility effects and stalling are ignored
- We have taken constant Taper value
- Pitch is varied according to expressions(more in approach part)
- We have taken Total Drag as 120% of fuselage Drag because we haven't calculated the tail and blade drag value
- Rotor baldes are nearly symmetric airfoil
-

3 Approach, Theory and Implementation

1. Started the program with defining parameters values which are constant for whole program such as Weight of helicopter, Number of blades, R, Rrc, RPM, density, Taper value, a, plate area etc.

2. Then we have calculated the Total Drag, Alpha TPP using the formula below:
   $Drag = 1.2 * 0.5 * \rho * V^2 * A * C_d$
   (1.2 is here for extra 20% drag due to tail and blade)
   $tan(\alpha_{TPP}) = \frac{Drag}{Weight}$
   in above equations:
   $C_d = 0.8$, $V = 73m/s$, $A = 6.8m^2$ and $Weight = 8000Kg$

3. Now we discretize the azimuth(ψ) and rotor blade. For discretization we have used 2000 points for rotor blade and 360 points for azimuth. Store these values in dr and dpsi for rotor blade and ψ respectively.

4. We have calculated helicopter thrust and C_T using tip path plane angle(α_{TPP} using the expressions below:
   $Thrust = \frac{Mass*g}{Cos(\alpha_{TPP})}$
   $C_T = \frac{Thrust}{\rho*A*(\Omega*R)^2}$
   From here we got: $C_T = 0.01011437$

5. Now we will use the Glauert Model for finding the final total non-uniform inflow ratio(λ), inflow part found using Glauert's method($\lambda_{i,G}$), λ_G as described below:

- First we have written the code for finding the $\lambda_{i,G}$, for which we have used iterative approach.
  first we have assume
  $\lambda_{i,G} = \frac{C_T}{2*\mu}$
  Then this value is used for calculate $\lambda_{i,G}$ using below formula till convergence upto 8 decimal digit:
  $\lambda_{i,G} = \frac{C_T}{2*\sqrt{\mu^2+(\mu*tan(\alpha)+\lambda_{i,G})^2}}$
  Code snippet

```
def find_lamda_ig():
    tmp_lamda_ig = CT/(2*mu)
    err = 100

    while err > 10**-8:
        lamda_ig = CT/(2*np.sqrt(mu**2 + (mu*np.tan(Alpha_tpp) + tmp_lamda_ig)**2))
        err = abs(tmp_lamda_ig-lamda_ig)
        tmp_lamda_ig = lamda_ig

    return tmp_lamda_ig
```

- Now we will calculate the value of λ_G using the expression :
  $\lambda_G = \lambda_{i,G} + \frac{V*Sin(\alpha)}{\Omega*R}$
- Using this λ_G we will now. calculate the ratio of λ_i and $\lambda_{i,G}$ using the formula below:

  $\frac{\lambda_i}{\lambda_{i,G}} = 1 + \frac{4/3\frac{\mu}{\lambda_G}}{1.2+\frac{\mu}{\lambda_G}} * \frac{r}{R} * cos(\psi)$
  Code Snippet

```
def find_lamda_G():
    tmp = (V*np.sin(Alpha_tpp))/(omega*R)
    return lamda_ig+tmp
```

```
lamda_G = find_lamda_G()
```

```
@numba.njit
def find_ratio_i_ig(r,psi):
    tmp = ((4*mu)/(3*lamda_G)/(1.2+mu/lamda_G))*(r*np.cos(psi)/R)
    return 1+tmp
```

- Finally after finding $\frac{\lambda_i}{\lambda_{i,G}}$ we will simply find the total non-uniform inflow ratio(λ) using the expression below:
  $\lambda = \lambda_i + \frac{V*sin(\alpha)}{\Omega*R}$
  Code snippet

```
#For total inflow
@numba.njit
def find_lamda(r,psi):
    tmp_a = (V*np.sin(Alpha_tpp))/(omega*R)
    tmp_b = lamda_ig*find_ratio_i_ig(r,psi)
    return tmp_a + tmp_b
```

6. Now I have written the functions for calculating the pitch and Flap at given r and ψ value.
   Before this we have define the variables θ_0,θ_{tw}, θ_{1s}, θ_{1c}, β_0, β_{1c}, β_{1s}, These parameters are we have to calculate at trim conditions and we have used the same name throughout the assignment.
   Now we calculate the pitch and Flap as below:
   $Pitch(\theta) = \theta_0 + \theta_{tw} * \frac{r}{R} + \theta_{1s} * sin(\psi) + \theta_{1c} * cos(\psi)$

   $Flap(\beta) = \beta_0 + \beta_{1c} * cos(\psi) + \beta_{1s} * sin(\psi)$

7. Now we define the function for calculating the value of angle of attack at given r and ψ. For calculating angle of attack we need to U_T,U_P, Pitch at given r and ψ. Expressions are as follows:
   $U_T = \Omega * r + V * cos(\alpha) * sin(\psi)$
   $U_P = (V * sin(\alpha) + v) * cos(\beta) + V * cos(\alpha) * sin(\beta) * cos(\psi)$
   Now for given r and ψ
   $Angle\ of\ Attack(r, \psi) = Pitch(r, \psi) - atan(\frac{U_P}{U_T})$
   Code Snippet:

3.3 Angle of Attack

```
@numba.njit
def angle_of_attack(r,psi,beta_0, beta_1c,beta_1s,theta_0, theta_1c, theta_1s,theta_tw):
    UT = omega*r + V*np.cos(Alpha_tpp)*np.sin(psi)
    UP = find_lamda(r,psi)*omega*R*np.cos(Flap(psi,beta_0, beta_1c,beta_1s)) + V*np.cos(Alpha_tpp)*np.sin(Flap(psi,b
    theta = pitch(r,psi,theta_0, theta_1c, theta_1s,theta_tw)
    return theta - np.arctan(UP/UT)
```

8. Now after calculating we will use following expressions to calculate Thrust, Pitching Moment, Rolling moment:
$C_l = a * Angle\ of\ \ Attack(r, \psi)$
$L^{'} = 0.5 * C_l * C * (U_T * *2 + U_P * *2) * sign(U_T)$
$Thrust = b * [\frac{1}{2\pi} \int_0^{2\pi} (\int_{Rrc}^{R} L^{'}\, dr)\, d\psi]$
$RollingMoment = b * [\frac{1}{2\pi} \int_0^{2\pi} (\int_{Rrc}^{R} L^{'} * r * sin(\psi)\, dr)\, d\psi]$
$RollingMoment = b * [\frac{1}{2\pi} \int_0^{2\pi} (\int_{Rrc}^{R} L^{'} * r * cos(\psi)\, dr)\, d\psi]$
Here a=5.8
b=number of blades
$L^{'}$ is sectional lift per unit span(sign function for taking care of reverse flow conditions)
For implementation we have first write the function for calculating $L^{'}$ at given r and ψ value using the expression above.
Code Snippet

```
@numba.njit
def L_prime(r,psi,beta_0, beta_1c,beta_1s,theta_0, theta_1c, theta_1s,theta_tw):
    Cl = a*angle_of_attack(r,psi,beta_0, beta_1c,beta_1s,theta_0, theta_1c, theta_1s,theta_tw)
    UT = omega*r + V*np.cos(Alpha_tpp)*np.sin(psi)
    UP = find_lamda(r,psi)*omega*R*np.cos(Flap(psi,beta_0, beta_1c,beta_1s)) + V*np.cos(Alpha_tpp)*np.sin(Flap(psi,b
    #print(Flap(psi))
    ans = 0.5*rho*C*Cl*(UT**2 + UP**2)*np.sign(UT)
    return ans
```

9. After calculating $L^{'}$ as described above now we have calculated the Thrust, Rolling Moment, Pitching Moment using the Integration. For integration we have used Trapezoidal rule, which is as follows:
$\int_a^b f(x)\, dx = \frac{\Delta x}{2} * [f(a) + f(b) + 2 * \sum_1^n f(x_i)]$
We used this above expression for calculating the integrals. Here for Thrust, Pitching Moment, Rolling moment we have double integral, so we have first calculate the integral by varying only one variable and keeping other one fixed. So we got list of integral calculated at every point and then we have used again Trapezoidal and calculate the Final integral.
Code Snippet

```
#Total Thrust:
@numba.njit
def find_Thrust(beta_0, beta_1c,beta_1s,theta_0, theta_1c, theta_1s,theta_tw):
    thrust_dr = np.zeros(len(dr))
    for i in range(len(dr)):
        thrust_dpsi = np.zeros(len(dpsi))
        for j in range(len(dpsi)):
            thrust_dpsi[j] = L_prime(dr[i],dpsi[j],beta_0, beta_1c,beta_1s,theta_0, theta_1c, theta_1s,theta_tw)
        thrust_dr[i] = ((2*np.pi)/(len(dpsi)*2))*(thrust_dpsi[0]+thrust_dpsi[-1]+2*np.sum(thrust_dpsi[1:-1]))

    Total_Thrust = ((R-Rrc)/(len(dr)*2))*(thrust_dr[0]+thrust_dr[-1]+2*np.sum(thrust_dr[1:-1]))
    return Total_Thrust
```

```
Total_Thrust = blades*find_Thrust(beta_0, beta_1c,beta_1s,theta_0, theta_1c, theta_1s,theta_tw)/(2*np.pi)
CT_integrate = Total_Thrust/(rho*(omega**2)*(R**4)*np.pi)
print('Total_Thrust: ',Total_Thrust)
print('CT_integrate: ',CT_integrate)
```

10. Now we have used the following expressions to calculate the C_T,$C_{m,roll}$, $C_{m,pitch}$:

$$C_T = \frac{Thrust}{2*\rho\pi R^4\Omega^2}$$

$$C_{m,roll} = \frac{Rolling\ \ Moment}{\rho*A*R(\Omega*R)^2}$$

$$C_{m,pitch} = \frac{Pitching\ \ Moment}{\rho*A*R(\Omega*R)^2}$$

```
#Rolling moment
@numba.njit
def find_rolling_moment(beta_0, beta_1c,beta_1s,theta_0, theta_1c, theta_1s,theta_tw):
    rolling_dr = np.zeros(len(dr))
    for i in range(len(dr)):
        rolling_dpsi = np.zeros(len(dpsi))
        for j in range(len(dpsi)):
            rolling_dpsi[j] = L_prime(dr[i],dpsi[j],beta_0, beta_1c,beta_1s,theta_0, theta_1c, theta_1s,theta_tw)*dr
        rolling_dr[i] = ((2*np.pi)/(len(dpsi)*2))*(rolling_dpsi[0]+rolling_dpsi[-1]+2*np.sum(rolling_dpsi[1:-1]))

    Total_rolling_moment = ((R-Rrc)/(len(dr)*2))*(rolling_dr[0]+rolling_dr[-1]+2*np.sum(rolling_dr[1:-1]))
    return Total_rolling_moment
```

```
Rolling_moment = blades*find_rolling_moment(beta_0, beta_1c,beta_1s,theta_0, theta_1c, theta_1s,theta_tw)/(2*np.pi)
C_rolling = Rolling_moment/(rho*(omega**2)*(R**5)*np.pi)
print('Rolling Moment: ',Rolling_moment)
print('Cm_rolling: ',C_rolling)
```

```
#Pitching Moment:
@numba.njit
def find_pitching_moment(beta_0, beta_1c,beta_1s,theta_0, theta_1c, theta_1s,theta_tw):
    pitching_dr = np.zeros(len(dr))
    for i in range(len(dr)):
        pitching_dpsi = np.zeros(len(dpsi))
        for j in range(len(dpsi)):
            pitching_dpsi[j] = L_prime(dr[i],dpsi[j],beta_0, beta_1c,beta_1s,theta_0, theta_1c, theta_1s,theta_tw)*d
        pitching_dr[i] = ((2*np.pi)/(len(dpsi)*2))*(pitching_dpsi[0]+pitching_dpsi[-1]+2*np.sum(pitching_dpsi[1:-1])

    Total_pitching_moment = ((R-Rrc)/(len(dr)*2))*(pitching_dr[0]+pitching_dr[-1]+2*np.sum(pitching_dr[1:-1]))
    return Total_pitching_moment
```

```
Pitching_moment = blades*find_pitching_moment(beta_0, beta_1c,beta_1s,theta_0, theta_1c, theta_1s,theta_tw)/(2*np.pi
C_pitching = Pitching_moment/(rho*(omega**2)*(R**5)*np.pi)
print('Pitching Moment: ',Pitching_moment)
print('C_pitching: ',C_pitching)
```

4 Simulator output and discussion

4.1 Trim values for the flight condition

4.1.1 Trim values of controls

θ_0	θ_{1c}	θ_{1s}	θ_{tw}
18.7	2.1	-11.2	-0.3

β_0	β_{1c}	β_{1s}	$\beta_{1c} - \theta_{1s}$	$\beta_{1s} - \theta_{1c}$
3	-1.07	1.68	-10.13	-0.42

4.1.2 Net Thrust, Pitching Moment and Rolling Moment at Trim

Thrust	C_T	Pitching Moment	$C_{m,pitch}$	Rolling Moment	$C_{m,roll}$
83162.493528	0.0103491	13.289264	2.265460e-07	-34.806567	-5.933580e-07

4.1.3 Tables showing what control variable is linked to what output

1. β_O **vs** C_T, $C_{m,pitch}$, $C_{m,roll}$

β_0	C_T	$C_{m,pitch}$	$C_{m,roll}$
1.0	0.009552	0.000472	-0.000229
2.0	0.009566	0.000193	-0.000225
3.0	0.009582	-0.000086	-0.000221
4.0	0.009600	-0.000365	-0.000215
5.0	0.009622	-0.000643	-0.000208
6.0	0.009646	-0.000922	-0.000201

2. $\beta_1 c$ **vs** C_T, $C_{m,pitch}$, $C_{m,roll}$

β_{1c}	C_T	$C_{m,pitch}$	$C_{m,roll}$
-1.0	0.009155	-0.000079	-0.000258
-2.0	0.009582	-0.000086	-0.000221
-3.0	0.010009	-0.000093	-0.000183
-4.0	0.010437	-0.000101	-0.000146
-5.0	0.010867	-0.000108	-0.000108
-6.0	0.011297	-0.000115	-0.000070

3. β_{1s} **vs** C_T, $C_{m,pitch}$, $C_{m,roll}$

β_{1s}	C_T	$C_{m,pitch}$	$C_{m,roll}$
1.0	0.009573	-0.000049	-0.000226
2.0	0.009582	-0.000086	-0.000221
3.0	0.009592	-0.000123	-0.000214
4.0	0.009604	-0.000160	-0.000208
5.0	0.009617	-0.000197	-0.000200
6.0	0.009632	-0.000234	-0.000192

4. θ_0 **vs** C_T, $C_{m,pitch}$, $C_{m,roll}$

θ_0	C_T	$C_{m,pitch}$	$C_{m,roll}$
15.0	0.003853	-0.000095	-0.001945
16.0	0.005762	-0.000092	-0.001370
17.0	0.007672	-0.000089	-0.000795
18.0	0.009582	-0.000086	-0.000221
19.0	0.011491	-0.000083	0.000354
20.0	0.013401	-0.000080	0.000929

5. θ_{tw} **vs** C_T, $C_{m,pitch}$, $C_{m,roll}$

θ_{tw}	C_T	$C_{m,pitch}$	$C_{m,roll}$
-0.1	0.009858	-0.000086	-0.000134
-0.2	0.009720	-0.000086	-0.000177
-0.3	0.009582	-0.000086	-0.000221
-0.4	0.009443	-0.000086	-0.000264
-0.5	0.009305	-0.000086	-0.000307
-0.6	0.009167	-0.000087	-0.000350

6. θ_{1c} **vs** C_T, $C_{m,pitch}$, $C_{m,roll}$

θ_{1c}	C_T	$C_{m,pitch}$	$C_{m,roll}$
1.0	0.009576	-0.000738	-0.000221
2.0	0.009582	-0.000086	-0.000221
3.0	0.009588	0.000566	-0.000220
4.0	0.009594	0.001218	-0.000219
5.0	0.009600	0.001870	-0.000218
6.0	0.009606	0.002522	-0.000217

7. θ_{1s} **vs** C_T**,** $C_{m,pitch}$**,** $C_{m,roll}$

θ_{1s}	C_T	$C_{m,pitch}$	$C_{m,roll}$
-9.0	0.011300	-0.000084	0.001239
-10.0	0.010441	-0.000085	0.000509
-11.0	0.009582	-0.000086	-0.000221
-12.0	0.008722	-0.000087	-0.000950
-13.0	0.007863	-0.000088	-0.001680
-14.0	0.007004	-0.000089	-0.002410

4.1.4 Observations and Discussion on above tables

1. **How we have selected the values mentioned in above tables:**

   - We have first calculated the required Thrust for Stable flight for parameters given in section 2. We got C_T value equal to 0.01011437.
   - Now we have to select the values of θ_0,θ_{tw}, θ_{1s}, θ_{1c}, β_0, β_{1c}, β_{1s} such that the required Thrust or C_T value is approximately equal to above calculated value or slightly higher than that. $C_{m,pitch}$ $C_{m,roll}$ we want close to zero.
   - Value of β_0 we have taken from 1 to 6 degree. We have taken positive value because the motion of helicopter is in forward stable flight and if calculate the coning angle value from the equation of coning angle we will get positive value of coning angle value because we can coning angle depends on the sectional lift per unit span generated over the area covered by the blade and radial distance and radial distance multiplied to it. So it is positive value and we got around 2-3 degree.
   - The values of β_{1c} taken as negative and in range -1 to -6. This is because we will get small U_P and large Angle of attack values for azimuth angle ranging from -90^o to 90^o compared to 90^o to 270^o which is desired for stable pitching moment.
   - Similarly the value of β_{1s} taken as positive from 1 to 6. This is because we need to reduce the effective angle of attack from 0 to 180 degree azimuth and increase from 180 to 360 degree azimuth for attaining the stable rolling moment.
   - Value of θ_0 is chosen in range 15-20 degree because in this range the C_T value we get is in out desired range for our forward velocity requirement.

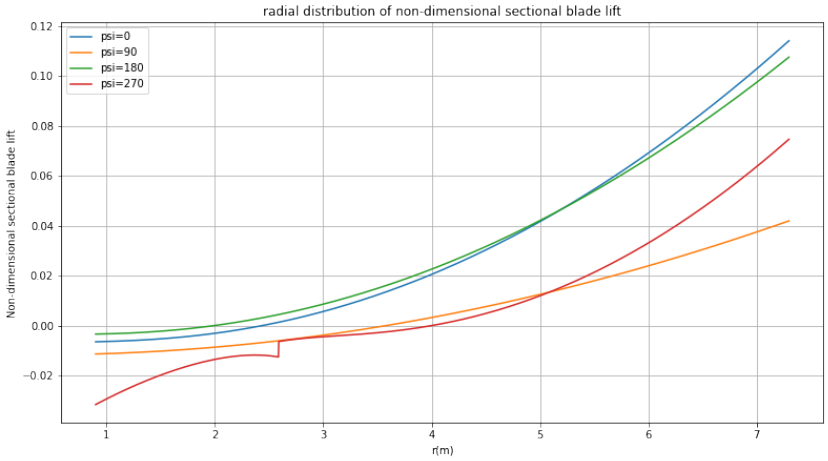
- The value of θ_{tw} is we have chosen negative because we need pitch to reduce as r increases to reduce the encountered profile drag on the blades. we have chosen value in the range -0.1 to -0.6 so that at tip effective angle of attack will remain positive.
- We have chosen the values of θ_{1c} positive. This is because as discussed in the class and slides the value of lift generated over azimuth 90 to 270 degree is high compared to azimuth 270 to 90 degree due to high blade flap and unstable pitching moment. In the equation of pitch we know that the θ_{1c} is coefficient of $cos(\psi)$ so if we take positive values of θ_{1c} then it will reduce the pitch values over azimuth 90 to 270 degree and increase over azimuth the 270 to 90 degree hence stabilize the pitching moment.
- In previous point we have seen that value of θ_{1c} is used taken positive to stabilize the pitching moment similarly the values of θ_{1s} is chosen negative to stabilize the Rolling moment. This can be explained by unstable rolling moment and low lift for azimuth 180 to 360 degree due to reverse flow. To overcome this we need high pitch values which can be achieved by the negative values of θ_{1s} because the value of $sin(\psi)$ is negative for ψ 180 to 360 range.

2. **Control Variable linked with Outputs(C_T,$C_{m,pitch}$, $C_{m,roll}$) and Choosing Trim value of controls**

   - Below observations are from Tables in section 4.1.3
   - We can see that change in θ_{1c} is will lead to high change in $C_{m,pitch}$ but very low change in $C_{m,roll}$. Means $C_{m,pitch}$ is very sensitive to θ_{1c} while $C_{m,roll}$ in not.
   - Opposite to above the change θ_{1s} will lead to high change in $C_{m,roll}$ while very low change in $C_{m,pitch}$
   - $\beta 1c$ have medium effect in changing the values of $C_{m,pitch}$ and $C_{m,roll}$ and between these two $C_{m,roll}$ is bit more sensitive to $\beta 1c$. Using this observation we can get the minimum value of roll.
   - Change in $\beta 1s$ leads to small change in value of $C_{m,roll}$ but medium change in $C_{m,pitch}$. Using this observation we can get the minimum value of pitch.
   - As mentioned earlier we have chosen θ_0 such that the value of C_T using the double integral and drag estimation will be same.
   - Using these observations we have reached at trim condition. We have chosen value to get C_T similar to calculated using drag estimation and the rolling moment and pitching moment close to zero.

4.2 Radial distribution of non-dimensional sectional blade lift

1. **Plot**



2. **Observations:**

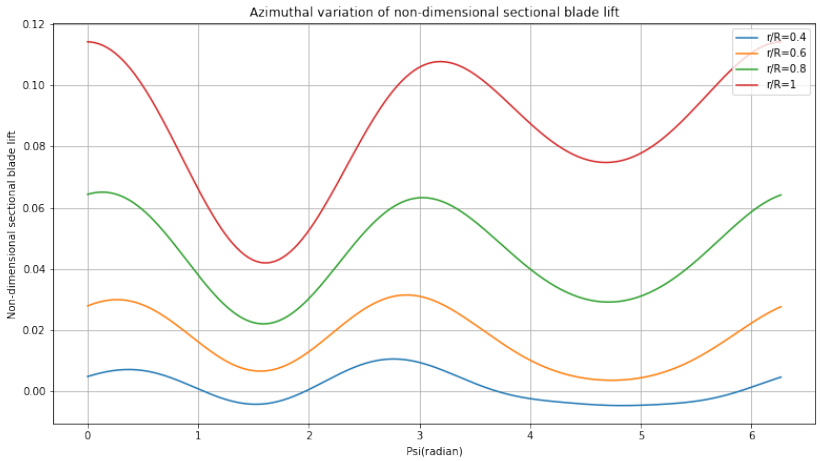
- We have used the trim values of controls for calculation of lift.
- We have calculated the value of Non-Dimensional Sectional Lift per unit Span at azimuth angles of 0°,90°,180°,270° and plotted radial distribution.
- As discussed earlier we have chosen the trim values such that the rolling moment and pitching moment are close to zero So non-Dimensional Sectional Lift per unit Span for *psi* value 0° and 180° are approximately equal.
- for *psi* value 0° and 180° the $\beta 1s$ and $\theta 1s$ does not affect the value of sectional lift as $sin(\psi)$ at these azimuth is zero.
- We have chosen the value of θ_{1s} negative to stabilize the Rolling Moment due to this the coefficient for $sin(\psi)$ at azimuth angle of 90° will be negative hence the pitch in reduced so lift will also reduced compared to azimuth of 0° and 180°.
- As r increases the value of U_T increases and $atan(U_P/U_T)$ decreases also pitch value is increases with increase in r. so effective angle of attack increases so the non-Dimensional Sectional Lift per unit Span.
- In case of azimuth 270° there is sudden change in in plot which is because of reverse flow. At 270° the $sin(\psi)$ is negative so U_T

is negative hence the value of effective angle of attack is positive. After some increase in r the value of U_T increase become zero and then positive so there is change in flow and change in sign of effective angle of attack so there is the sudden change in curve for azimuth angle 270°.

- Close to root the value of sectional lift is negative which is due to low value of U_T which to negative angle of attack hence the negative lift.

4.3 Azimuthal variation of non-dimensional sectional blade lift

1. **Plot**



2. **Observations:**

- Similar to previous we have used the trim values of controls for calculation of lift.
- We have plotted the azimuthal variation of non-dimensional sectional blade lift at r/R= 0.4, 0.6, 0.8 and 1.0.
- As we increase the r/R value the curve is shifting above due to increase in the value of $U_T^2 + U_P^2$.
- We have also not estimated the effect of tip loss so the value of lift increases also due to that reason too.
- We can see that for r/R=0.4 the value of non-dimensional sectional blade lift is negative at 90° and 270° because the effective angle of attack at these small value of r is negative due to small value of $\Omega * r$ hence negative lift.

- Non-dimensional sectional blade lift Curves for all r/R values is sinusoidal for given r/R value with minima at 90° and 270° and maxima at 0° and 180°.
- Minima and Maxima are because we have chosen the control value for minimizing the pitching and rolling moment. If we look closely at expressions of pitch, flap and angle it is we can see that from 0° to 90° there is major role of θ_{1s} as it is negative as well as large in magnitude so the value pitch reduces as $sin(\psi)$ is positive so product in negative so lift is reducing till 90° from 90° to 180° the reverse phenomena occurs hence the value start increase. this same will repeat after the 180° azimuth angle but peak is slightly reduces due to θ_{1c} term.
- Here the effect of β_{1s} and β_{1c} not clearly visible due to small value of these parameters.

References

[1] `https://fas.org/man/dod-101/sys/ac/ah-64.htm`

[2] `https://www.army-technology.com/projects/apache/`

[3] `https://en.wikipedia.org/wiki/Boeing_AH-64_Apache`

[4] `https://www.boeing.com/defense/ah-64-apache/#/performance`

[5] Slides from Course AE667: Prof. Dhwanil Shukla