AE667-Assignment4

Manoj Bhadu 170010036

April 2021

Contents

1 Literature survey

1.1 Data Used

Parameters	Value
Density at sea level(Kg/m^3)	1.225
Weight(Kg)	8000
R	5.5
RPM	383.36
Number of blades	4
a	5.8
C_{d0}	0.010
Equivalent Plate area	1.95096
Chord(m)	0.312
Rotor weight(kg)	36.3
Tail Rotor weight(kg)	2.49
Tail rotor radius(m)	0.981
Tail rotor Chord	0.12
Distance of tail from C.O.G.(m)	7.1
Max Continuous Power(kW)	632
Take off Power(kW)	667
full tank fuel	694
Recommended Cruise speed(m/s)	67.5
Fuel consumption at Recommended Cruise speed(kg/h)	251
Fuel consumption at 65 KIAS(kg/h)	197
Power Factor(K)	1.14

2 Assumption

- We have used blade element momentum theory for calculating the Rotor power
- We have calculated the Range, Endurance etc quantities at height 420m above the ground
- Density is 1.21 at 420m above ground
- The assumptions made in BEM theory are valid here to
- We have not calculated the value of fuselage down wash due to data unavailability
- compressibility effects and stalling are ignored
- We have taken constant chord length

- We have chosen the value of C_{d0}, a, Power Factor, installation loses according to some other helicopter data available.

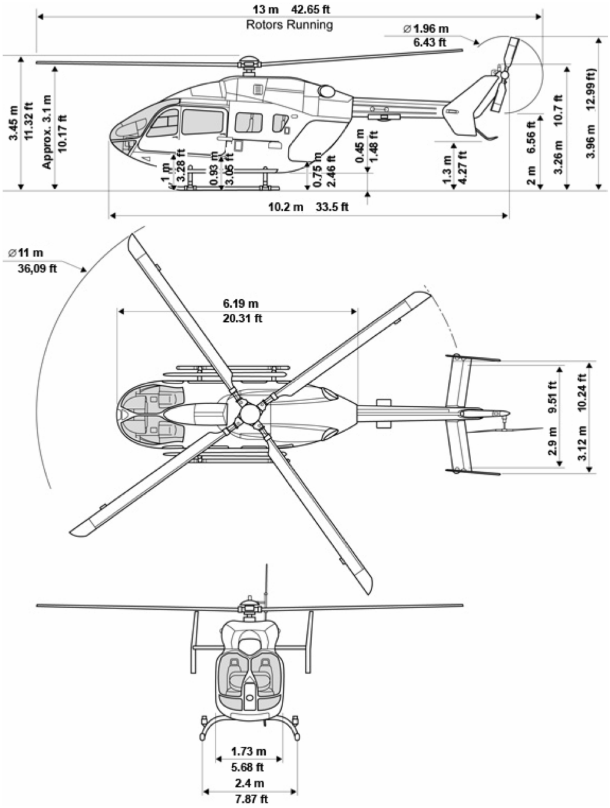

Figure 1: Dimension of helicopter

3 Approach, Theory and Implementation

3.1 Parameters defining

First we have defined the parameters such as ρ, Gross weight, blades, rpm, omega, Chord length, Radius and some of tail rotor dimensions.

3.2 Service and Absolute ceiling

- First we have calculated the Thrust value from weight
- Calculated the solidity as $\sigma = \frac{b*C}{\pi R}$
- Taken K as 1.14 and installation loss as 0.97

- Now we have calculated the Tail rotor power
  $Tail\ rotor\ Power = Tail\ induced\ Power + Tail\ Profile\ Power$
  Where $Tail\ induced\ power = Tail\ Thrust * \sqrt{\frac{Tail\ Thrust}{2\rho A}}$
  $Tail\ Profile\ Power = (tail\ solidity * Cd0 * \rho * tail\ omega^3 * tail\ radius^5 * \pi)/8$
  Code Snippet

```
#Calculating the Tail rotor power(includes tail induced and profile power)
def Tail_rotor_power(h, Thrust):
    rho_ratio = ((1-(0.00198*h/288.16))**4.2553)
    Torque = Power_h_required_service(h,Thrust)/omega
    tail_thrust = Torque/tail_COG
    tail_induced_power = tail_thrust*np.sqrt(tail_thrust/(2*rho*rho_ratio*np.pi*tail_radius*tail_radius))
    tail_profile_power = (tail_solidity*Cd_0*rho_ratio*rho*(tail_omega**3)*(tail_radius**5)*np.pi)/8
    tail_power = tail_induced_power+tail_profile_power
    return tail_power
```

- Now we have calculated the power available at height h.

- for above calculation we have used:
  $\frac{P}{P_{sealevel}} = \frac{\rho}{\rho_{sealevel}}$
  where: $\frac{\rho}{\rho_{sealevel}} = [1 - \frac{0.00198*h}{288.16}]^{4.2553}$

  Code Snippet

```
#Calculate the power available at height h
def Power_h_available(h,power):
    ratio = ((1-(0.00198*h/288.16))**4.2553)
    power_at_h = power*ratio*f
    return power_at_h
```

- Now we have calculated the value of Power Required by main motor using following expression :
  $P = K * T * \sqrt{\frac{T}{2\rho A}} + (\sigma * Cd_0 * \rho * \Omega^3 *^5 * \pi)/8$

Code Snippet

```
#Power required by main rotor including induced, profile power
def Power_h_required_service(h,Thrust):
    rho_ratio = ((1-(0.00198*h/288.16))**4.2553)
    induced_power = (Thrust*np.sqrt(Thrust/(2*rho*rho_ratio*np.pi*R*R)))/1.06
    profile_power = (Solidity*Cd_0*rho_ratio*rho*(omega**3)*(R**5)*np.pi)/8

    power = induced_power + profile_power
    #print(power)
    return power
```

- Now we have written the function for the calculation of Maximum climb rate.
- we know $max\ climb\ rate = \frac{excess\ power}{Gross\ weight}$
  where: Excess Power = Power available - Power required
  Code Snippet

```
: #Max climb rate at height h for given max power and Thrust
def max_climb_rate(h,power_installed, Thrust):
    power_available = Power_h_available(h,power_installed)
    power_required = Power_h_required_service(h, Thrust)+Tail_rotor_power(h,Thrust)
    climb_rate = (power_available-power_required)/(9.81*weight)
    return climb_rate
```

- Now we have tried for different values of height which are as shown in figure below
- From below fig we can see that the absolute ceiling is 5720m and service ceiling is 4900m.

3.3 Maximum Forward Velocity

- We have calculated the maximum forward speed at height 420m
- For Calculating the maximum forward velocity we have used the following expression :
  $P_{available} = Power_{required}$
  $P_{available} = P_{induced} + P_{profile} + P_{fuselage\ Drag}$
  $P_{fuselage\ Drag} = D * V_{max}$
  $D = 0.5 * \rho * equivalent\ plate\ area * V_{max}^2$
  Using above four equations we have calculated the V_{max} for h=420m.
  Code Snippet

```
]: height = np.arange(4700,6000,50)
   for i in range(len(height)):
       max_climbrate = max_climb_rate(height[i],max_cont_power,Thrust )
       print(height[i], max_climbrate)
```

```
4700 0.6290161145240898
4750 0.598728152963249
4800 0.5684509101083103
4850 0.5381843459376742
4900 0.5079284203912867
4950 0.4776830933705174
5000 0.4474483247380186
5050 0.41722407431765546
5100 0.38701030189431557
5150 0.3568069672138296
5200 0.3266140299828024
5250 0.29643144986853925
5300 0.2662591864988355
5350 0.23609719946194413
5400 0.20594544830638176
5450 0.17580389254083134
5500 0.14567249163398405
5550 0.11555120501443852
5600 0.08543999207056971
5650 0.05533881215033816
5700 0.02524762456126493
5750 -0.004833611429791402
5800 -0.034904936596719234
5850 -0.06496639175430859
5900 -0.09501801775836437
5950 -0.1250598555058466
```

```
']: print("Absolute Ceiling:",5720)
    print("service Ceiling:",4900)
```

```
Absolute Ceiling: 5720
service Ceiling: 4900
```

Figure 2: fig: Max Power for Different height

```
: #Max forward speed calculation
  #Includes the required power due main, tail rotor and parasite drag
  #Max speed is when power required and power available are equal
  def max_forward_speed(h, Thrust):
      rho_ratio = ((1-(0.00198*h/288.16))**4.2553)
      power_available = Power_h_available(h,max_cont_power)
      power_required = Power_h_required_service(h, Thrust) + Tail_rotor_power(h,Thrust)

      max_forward_speed = (((power_available - power_required)*2)/(rho*rho_ratio*plate_area*0.8))**(1/3)
      return max_forward_speed
```

```
: print("Max Forward Velocity: ", max_forward_speed(420, Thrust))
```

```
Max Forward Velocity:  48.196542270474026
```

Figure 3: fig:Code Snippet for max forward velocity

3.4 Range and Endurance

- We have calculated the range and endurance value at the height of 420m and cruise speed of 67.5m/s(which is recommended)
- Fuel tank capacity is 694kg
- Fuel consumption at recommended cruise is 234kg/hr
- Fuel consumption at 65 KAIS is 197kh/hr
- Now first we have calculated the value of permissible TOF which is $\frac{total\ fuel}{fuel\ consumption\ at\ recommended\ cruise}$
- In code you can see we have calculated the Total power required for recommended cruise speed. This is because for our new design we don't have the fuel consumption rate. So for calculating that we will calculate the Total Power required for new design and then using this power and fuel consumption rate we will calculate the fuel consumption for recommended cruise speed. Power calculations are same as in previous sections.
- Now we have calculated the Range and endurance as follows:

```
def Range():
    range_ = cruise_speed*permissable_TOF
    return range_

print("Range: ", Range())

Range:  720692.3076923076

def Endurance():
    endurance = total_fuel/fuel_consumption_65_kias
    return endurance

print("Endurance: ", Endurance())

Endurance:  12682.23350253807
```

Figure 4: fig:Range and Endurance

3.5 Autorotative index

- First we have calculated total inertia value for main rotor and tail rotor and then calculated total inertia value of helicopter
  $J_{rotor} = blades * (4 * R^2 + C^2) * rotor\ weight/12$
  $J_{tail} = 2 * (4 * R^2_{rotor} + C^2_{tail}) * tail\ weight/12$
  $J_{total} = J_{rotor} + J_{tail} * \frac{\Omega_{tail}}{\Omega_{rotor}}$
- Now Autorotative index can be calculated as:
  $AI = \frac{J_{total} * \Omega^3 * \rho * A}{GW * \rho_0 * T}$

Code Snippet

```python
#defining inertia parameters
rotor_weight = 36.3                                         #one ROTOR weight
J_rotor_blades = (blades*(4*R*R + C*C)*rotor_weight)/12     #inertia of all rotors
tail_rotor_weight = 2.49                                    #in Kg
J_tail_rotors = 2*((4*(tail_radius**2)+(tail_chors**2)))*tail_rotor_weight/12
J_total = J_rotor_blades + J_tail_rotors*tail_omega/omega

#Lock Number
lock_number = (rho*C*a*(R**4))/J_total
print("Lock Number:", lock_number)
```

```
Lock Number: 1.3134434385010927
```

```python
def AI():
    rho_h = 1.21
    Drag = 0.5*rho_h*plate_area*(cruise_speed**2)*0.8
    alpha_tpp = np.arctan(Drag/(weight*9.81))
    Thrust = (weight*9.81)/(np.cos(alpha_tpp))
    autorotative_index = (J_total*(omega**2)*rho_h*np.pi*(R**2))/(weight*rho*Thrust)
    return autorotative_index
```

```python
print("Autorotative Index: ",AI())
```

```
Autorotative Index:  2.0639030343987788
```

Figure 5: fig:Autorotative Index

4 New design and approach

- I have tried changing the different quantities and then choose the quantities which has given the absolute ceiling of 8008m.
- For new design first for a given height we have calculated the value of density
- After calculating the density we have calculated the value of available power
- Now we have available power so we need to find the parameters such that max climb rate is zero or in other words the power required is equal to power available.
- So for this we have varied the rotor radius from 5.2 to 5.8 m
- We have varied the gross weight from 3000 kg to 3500 kg
- We have varied the Chord length value from 0.280 to 0.325
- We have also tried varying the tail rotor parameters but they does not give that much difference in values
- After Varying all above mentioned parameters we have achieved at following value of parameters

Gross weight(kg)	Chord(m)	Rotor Radius(m)	tail rotor radius(m)
3100	0.297	5.498	0.981

5 Results and comparison:

5.1 EC145 Helicopter performance Computed vs Published

Quantity	Computed	Published
Absolute Ceiling(m)	5720	No Data
Service Ceiling(m)	4900	5540
Max forward speed(m/s)	48	69.44
Range(m)	720692.30	685000
Endurance(min)	211	220
Autorotative Index	2.0639	No Data

5.2 Helicopter performance computed New Helicopter vs EC145

Quantity	EC145	New Helicopter
Absolute Ceiling(m)	5720	8008
Service Ceiling(m)	4900	7200
Max forward speed(m/s)	48	53
Range(m)	720692.30	744948.27
Endurance(min)	211	183.93
Autorotative Index	2.0639	2.335

6 Discussions

- The value of service ceiling is differ from advertised value by approx 500m. This is because in the calculation of power required we have assumed constant chord length(in advertised value this is linear) and constant C_d. Also we doesn't account for the tip loss. We have also included the tail rotor power and for that we have taken 2-3 approximation so the discrepancy from there can be possible
- Autorotative index data is not available.
- Values of Max continuous power, take off power, GW also varies from reference to reference. That is mostly due to different manufacturing.
- The endurance values doesn't differ much from computed and published.
- Range values are differs by approx 35km.
- Values for the New designed helicopter and the computed are not much diverted from expected.

- We can see that there is difference in absolute ceiling so there should be difference in service ceiling and that is clearly visible
- Range values are similar because for Range calculation we have taken same values of cruise speed and the fuel consumption is calculated from EC145's computed Total Power to recommended fuel ratio.
- Range is significantly reduced, this is due to the the density variation with height. So the values of available power will be less at higher altitude hence less Endurance
- Autorotative index has increases because of the decrements in the Chord lenght and Gross weight as these quantities directly appears in formula.

References

[1] https://exclusiveaircraft.co.uk/sites/default/files/brochure/Eurocopter-EC145-Brochure.pdf

[2] https://www.homelandsecurity-technology.com/projects/ec145-t2-h145-twin-engine-multi-role-helicopter/

[3] https://en.wikipedia.org/wiki/Eurocopter_EC145

[4] https://people.clarkson.edu/~pmarzocc/AE429/AE-429-10.pdf

[5] https://dspace-erf.nlr.nl/xmlui/bitstream/handle/20.500.11881/1285/34.pdf?sequence=1

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