Author: Adrian Szklarski, 09.06.2022.
Technological level: Professional
Programming level: Mid+
Technologies used: Python 3.10 | PyCharm | Linux | OpenOffice

Description of the problem: A supersonic aircraft flies at an altitude $H[m]$, with a speed $w[\frac{m}{s}]$. Estimate the heat flux entering from $1m^2$ of wing surface, at a point $x[m]$ away from the leading edge whose temperature is $T_w[^oC]$. Parameters of the reference atmosphere according to the table.

Standard temperature up to 11.000m: $T=T_0-6{,}5\frac{K}{km}$

Standard pressure up to 11.000m: $p=p_0(1-\frac{H}{44300})^{5{,}256}$

Standard density up to 11.000m: $\rho=\rho_0(1-\frac{H}{44300})^{4{,}256}$

Speed of sound up to 11.000m: $a=a_0\sqrt{\frac{T}{288}}$

Speed of sound for an ideal gas: $a_\infty=\sqrt{kRT_\infty}$

Mach number: $Ma_\infty=\frac{w_\infty}{a_\infty}$

The viscosity of air is determined from Sutherland's formula:

$$\mu_\infty=17.21*10^{-6}*\left(\frac{1+\frac{122}{273}}{1+\frac{122}{273+T_\infty}}*\sqrt{\frac{(273+T_\infty)}{273}}\right)$$

Reynolds number: $Re_x=\frac{w_\infty x}{\mu_\infty/\rho_\infty}$

The ratio of the average free path of an air molecule to the characteristic dimension of the wing:

$$Kn=\frac{M_\infty a}{\sqrt{Re_x}}\ .$$

The phenomenon takes place at high velocity, so the energy dissipation phenomenon cannot be neglected in the boundary layer. Temperature recovery factor for turbulent motion: $p_{temp}={}^3(\sqrt{Pr})$

Adiabatic wall temperature: $T_{ad}=T_\infty+\frac{p_{temp}*w_\infty^2}{2}c_p$

Specific enthalpy: $i_{ad}=c_p[T_\infty+0{,}5(T_w-T_\infty)+0{,}22(T_{ad}-T_\infty)]$

Density, Reynolds number: $\rho=\frac{p}{RT}$, $Re_x=\frac{w_\infty x}{\mu/\rho}$

Stanton number and heat transfer coefficient: $S=\frac{f_x}{2Pr^{\frac{2}{3}}}$, $\alpha=Sw_\infty\rho$

The density of the heat flux entering the surface of the lobe: $q=\alpha\, c_p(T_{ad}-T_w)$

Code:

```
'''  Adrian Szklarski
      10.06.2022'''

import math

T0 = 288.15 # K
p0 = 1013.25 # hPa
ro0  = 1.2255  # kg/m³
ni0  = 1.461*10**-5 # m²/s
a0 = 340.3  # m/s

class StandardAtm:

    def __init__(self, T0, p0, ro0, ni0, a0, H):
        self.T0 = T0
        self.p0 = p0
        self.ro0 = ro0
        self.ni0 = ni0
        self.a0 = a0
        self.H = H

    def temp(self):
        try:
            self.T = round(self.T0 - 6.5*(self.H / 1000), 2)
        except ZeroDivisionError:
            self.T = self.T0
        return self.T

    def pres(self):
        self.p = round(self.p0*((1-(H/44300))**5.256), 2)
        return self.p

    def dens(self):
        self.ro = round(self.ro0 * ((1 - (H / 44300)) ** 4.256), 2)
        return self.ro

    def speed(self):
        self.a = round(self.a0*(SA.temp()/288)**0.5, 2)
        return self.a


class Wing(StandardAtm):

    def __init__(self, T0, p0, ro0, ni0, a0, H, a, x, cp, Pr):
        super().__init__(T0, p0, ro0, ni0, a0, H)
        self.a = a
        self.x = x
        self.cp = cp
        self.Pr = Pr


    def speed_of_sound(self):
        self.a_inf = math.sqrt(1.4*22730 / SA.dens())
        return round(self.a_inf, 2)

    def number_ma(self):
        self.Ma = self.a / self.a_inf
        return round(self.Ma, 2)

    def sutherland(self):
```

```
        self.su = 17.21*(10**-6) * (1+(122/273)) / (1+122/(273+SA.temp()-275.15)) * math.sqrt((SA.temp()-
275.15+273)/273)
        return round(self.su, 7)

    def number_re(self):
        self.R = self.a * self.x /(Wing.sutherland(self)/SA.dens())
        if self.R > 10**7:
            text = f'Reynolds number: the flow is turbulent '
        else:
            text = f'Reynolds number: the flow is laminar '
        return text, round(self.R, 2)

    def number_ku(self):
        self.Kn = Wing.number_ma(self)/math.sqrt(Wing.number_re(self)[1])
        if self.Kn < 0.01:
            text = f'continuous center'
        else:
            text = f'non-continuous center'
        return text, round(self.Kn, 6)

    def temp_recovery(self):
        self.pt = self.Pr**(1/3)
        return round(self.pt)

    def wall(self):
        self.Tad = SA.temp()-275.15 + Wing.temp_recovery(self)*(self.a**2/(2*self.cp*1000))
        return round(self.Tad, 3)

    def enthalpy(self):
        a = self.cp
        t = SA.temp()-275.15
        w = Wing.wall(self)
        self.e = a * (t + 0.5*(-5-t) + 0.22*(w-t)) # '-5' is Temperature at the leading edge
        return round(self.e, 2)

    # def __str__(self):
    #     return f"Speed of sound for an ideal gas: {Wing.speed_of_sound(self)}, \nMach number:
{Wing.number_ma(self)}, \n\
    #     The viscosity of air is determined from Sutherland's formula: {Wing.sutherland(self)}, \
n{Wing.number_re(self)} \n\
    #     Knudsen number: {Wing.number_ku(self)}, Temperature recovery: {Wing.temp_recovery(self)}, Adiabatic wall
temperature: {Wing.wall(self)}, \n\
    #     Specific enthalpy: {Wing.enthalpy(self)} "

class Resume(Wing):

    def __init__(self, T0, p0, ro0, ni0, a0, H, a, x, cp, Pr, t, dvc, p):
        super().__init__(T0, p0, ro0, ni0, a0, H, a, x, cp, Pr)
        self.t = t
        self.dvc = dvc
        self.p = p

    def dencity(self):
        ro = 22730 / (286 * (273 + self.t))
        Rex = self.a * self.x / (self.dvc / ro)
        St = 0.37 / (2 * math.pow(math.log(Rex), 2.584) * math.pow(self.p, (2/3)))/10
        alfa = St * self.a * ro/10
        self.q = alfa * self.a * (SA.temp() - (-5))  # '-5' is Temperature at the leading edge
        return self.q

    def __str__(self):
        return f'{Resume.dencity(self)}'
```

```
if __name__ == '__main__':

    while True:
        try:
            H = float(input('H[m]: '))
            a = float(input('v[m/s]: '))
            x = float(input('x[m]: '))
            cp = float(input('cp: '))
            Pr = float(input('Pr: '))
            if H >= 0 and a >= 340.3 and x >= 0:

                SA = StandardAtm(T0, p0, ro0, ni0, a0, H)
                Wings = Wing(T0, p0, ro0, ni0, a0, H, a, x, cp, Pr)
                # print(Wings)

                print('For', Wings.enthalpy(), 'from the tables give the temperature [C]: ')
                t = float(input(''))
                print('For', Wings.enthalpy(), 'from the tables give the dynamic viscosity coefficient [Pa s]: ')
                dvc = float(input(''))
                print('For', Wings.enthalpy(), 'from the tables give the number of Prandlt: ')
                p = float(input(''))

                R = Resume(T0, p0, ro0, ni0, a0, H, a, x, cp, Pr, t, dvc, p)
                print('The density of the heat flux entering the surface of the lobe: ', R.dencity())
                break
            else:
                print('Wrong value was entered')
        except ValueError:
            print('A bad value was entered that is not a number')
```

Resluts:

```
Python 3.10.4 (main, Apr  2 2022, 09:04:19) [GCC 11.2.0]
H[m]: >? 11000
v[m/s]: >? 800
x[m]: >? 1.5
cp: >? 1.005
Pr: >? 0.728
For 38.49 from the tables give the temperature [C]:
>? 33.2
For 38.49 from the tables give the dynamic viscosity coefficient [Pa s]:
>? 0.00001879
For 38.49 from the tables give the number of Prandlt:
>? 0.7
The density of the heat flux entering the surface of the lobe:  60.56
```