Author: Adrian Szklarski, 03.06.2022.
Technological level: Intermediate
Programming level: Mid
Technologies used: Python 3.10 | PyCharm | Linux | OpenOffice

Description of the problem: Calculate the aerodynamic coefficients for a rectangular airfoil with dimensions: wingspan w = 12 [m], wing chord c = 1,5 [m], aspect ratio $\Lambda = 5$.

α	-4^0	0^0	4^0	8^0	12^0	16^0	20^0
c_Z	- 0,05	0,24	0,51	0,78	1,05	1,30	1,40
c_X	0,015	0,017	0,030	0,055	0,087	0,198	0,202

Assuming a linear change of c_Z in the range of angles $\in < -4^0 \div 12^0 >$ we calculate the average value:

$$\frac{dC_z}{d\alpha}=\frac{C_Z^{12^\circ}-(-C_Z^{4^\circ})}{12^\circ+4^\circ}\cdot\frac{180}{\pi} \qquad [1]$$

then we calculate $\frac{\Lambda}{\frac{dC_Z}{d\alpha}}$ = 1.27 and then using the graph „ c_Z coefficient gradient change with aspect ratio Λ for a rectangular wing„ available in the general literature $\frac{\Lambda}{\frac{dC_Z}{d\alpha}}$ = f($\frac{\Lambda}{\left(\frac{dC_Z}{d\alpha}\right)_{\Lambda=\infty}}$) we determine the wing aspect ratio $\Lambda_2 = 8$.

From the above calculations from the chart correction coefficients τ, δ = f($\frac{\Lambda}{\left(\frac{dC_Z}{d\alpha}\right)_{\Lambda=\infty}}$) we get:

$$\frac{\Lambda}{\left(\frac{dC_Z}{d\alpha}\right)_{\Lambda=\infty}} \rightarrow \quad \tau_1 = 0.16, \quad \delta_1 = 0.04$$

$$\frac{\Lambda_2}{\left(\frac{dC_Z}{d\alpha}\right)_{\Lambda=\infty}} \rightarrow \quad \tau_2 = 0.21, \quad \delta_2 = 0.075$$

Change of attack angle:

$$\Delta\alpha=\alpha_2-\alpha_1=C_Z\left(\frac{\frac{1.21}{8}-\frac{1.16}{5}}{\pi}\right)=-0.026\,C_Z \qquad [2]$$

and change of drag:

$$\Delta C_X=\Delta C_{Xi}=C_Z^2\left(\frac{\frac{1.075}{8}-\frac{1.04}{5}}{\pi}\right)=-0.0235\,C_Z^2 \qquad [3]$$

Results of the analysis:

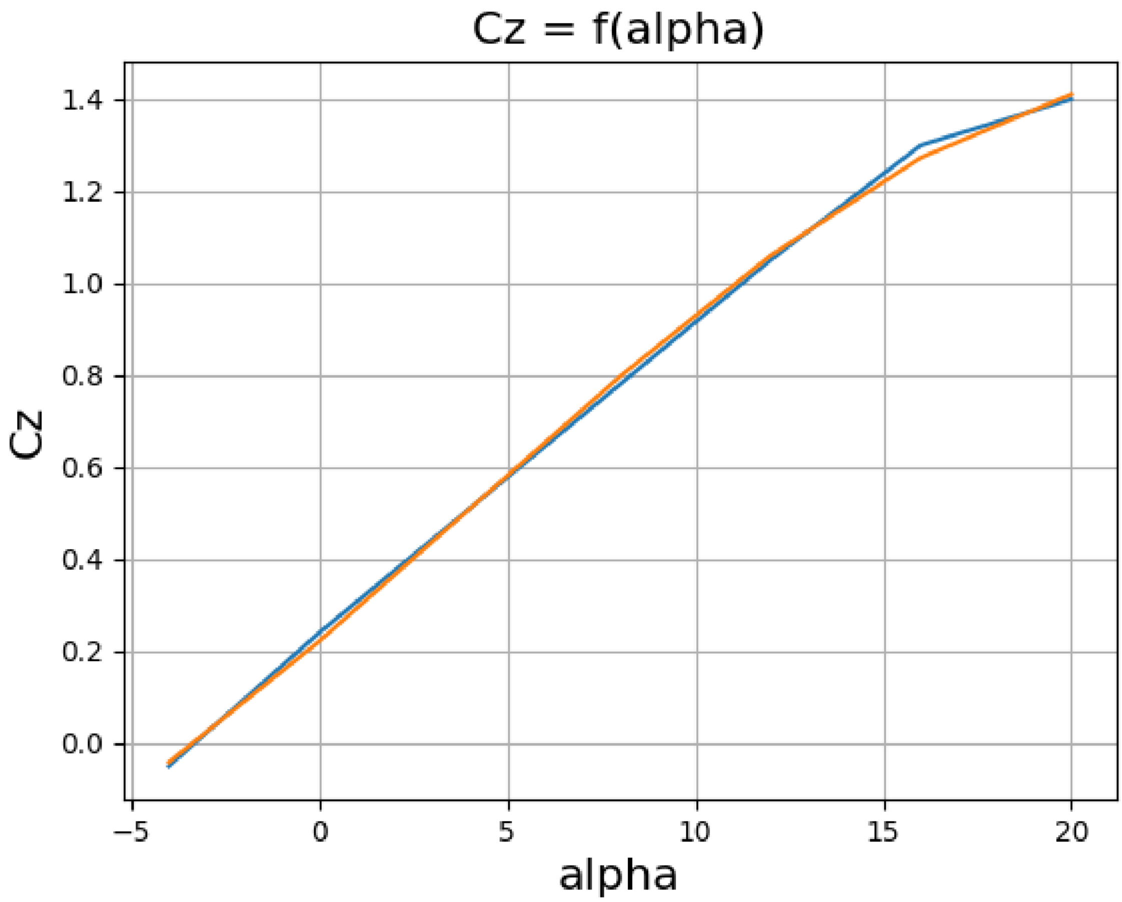
```
import matplotlib.pyplot as plt
from scipy.interpolate import splrep, splev
import numpy as np
import math


alpha = [-4, 0, 4, 8, 12, 16, 20]
Cz = [-0.05, 0.24, 0.51, 0.78, 1.05, 1.3, 1.4]
Cx = [0.015, 0.017, 0.03, 0.055, 0.087, 0.198, 0.202]

class Aero:


    def __init__(self, alpha, Cz, Cx):
        self.alpha = alpha
        self.Cx = Cx
        self.Cz = Cz

    def cz_cx(self):
        cz = np.array(self.Cz)
        cx = np.array(self.Cx)
        self.div = (cz / cx).round(5)
        return self.div

    def rad(self):
        self.deg_rad = np.array(self.alpha) * (math.pi / 180)
        return self.deg_rad.round(5)

    def delta_alpha(self):
        self.da = -0.026 * np.array(self.Cz)
        return self.da.round(5)

    def czarecz(self):
        self.Cz2 = np.array(self.Cz) * np.array(self.Cz)
        return self.Cz2.round(5)

    def deltaCxi(self):
        self.Cxi = (-0.0235 * np.array(self.Cz) * np.array(self.Cz))
        return self.Cxi.round(4)

    def Cx8(self):
        self.Cx8 = np.array(self.Cx)
        return self.Cx8.round(5)

    def CzCx8(self):
        self.CzCx = (np.array(self.Cz) / np.array(self.Cx))
        return self.CzCx.round(5)

    def __str__(self):
```

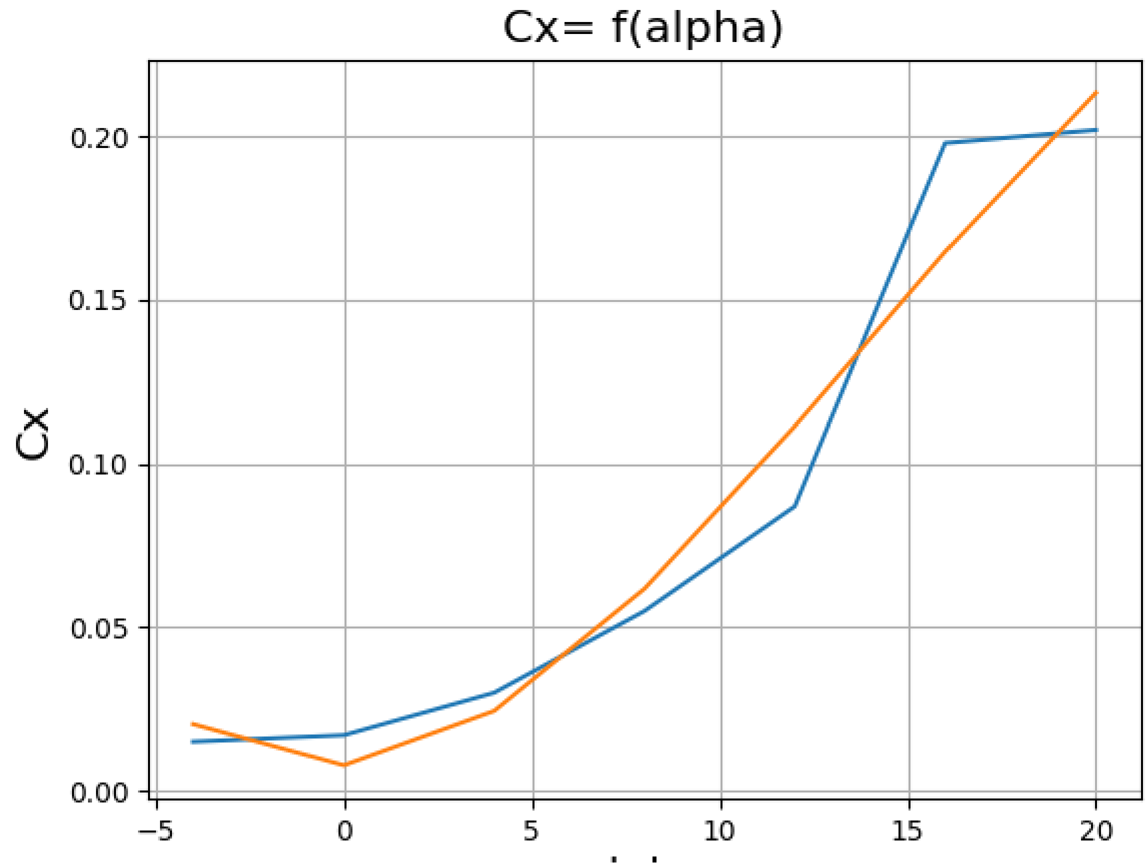
```
        return f'{self.div}, {self.deg_rad}, {self.da}, \
            {self.Cz2}, {self.Cxi}, {self.Cx8}, {self.CzCx8}'


if __name__ == '__main__':
    a = Aero(alpha, Cz, Cx)
    print(' Cz/Cx(5)   : ', a.cz_cx(), '\n alpha[rad] : ', a.rad(), '\n delta_alpha: ', a.delta_alpha(), '\n'
                                                                                      ' Cz^2       : ',
          a.czarecz(), '\n delta Cxi  : ', a.deltaCxi(), '\n Cx(8)      : ', Cx, '\n'
                                                                       ' Cz/Cx(8)   : ', a.CzCx8())

    # plot Cz = f(alpha)
    plt.title('Cz = f(alpha)', fontsize=16)
    bspl = splrep(alpha, Cz, s=5)
    bspl_y = splev(alpha, bspl)
    plt.plot(alpha, Cz)
    plt.plot(alpha, bspl_y)
    plt.xlabel('alpha', fontsize=16)
    plt.ylabel('Cz', fontsize=16)
    plt.grid(True)
    plt.show()

    # plot Cx = f(alpha)
    plt.title('Cx= f(alpha)', fontsize=16)
    bspl = splrep(alpha, Cx, s=5)
    bspl_y = splev(alpha, bspl)
    plt.plot(alpha, Cx)
    plt.plot(alpha, bspl_y)
    plt.xlabel('alpha', fontsize=16)
    plt.ylabel('Cx', fontsize=16)
    plt.grid(True)
    plt.show()

    # plot Cz = f(Cx)
    plt.title('Cz= f(Cx)', fontsize=16)
    bspl = splrep(Cx, Cz, s=5)
    bspl_y = splev(Cx, bspl)
    plt.plot(Cx, Cz)
    plt.plot(Cx, bspl_y)
    plt.xlabel('Cx', fontsize=16)
    plt.ylabel('Cz', fontsize=16)
    plt.grid(True)
    plt.show()

    # plot Cz/Cx = f(alpha)
    plt.title('Cz/Cx= f(alpha)', fontsize=16)
    bspl = splrep(alpha, a.CzCx8(), s=3)
    bspl_y = splev(alpha, bspl)
    plt.plot(alpha, a.CzCx8())
    plt.plot(alpha, bspl_y)
    plt.xlabel('alpha', fontsize=16)
    plt.ylabel('Cz/Cx', fontsize=16)
```

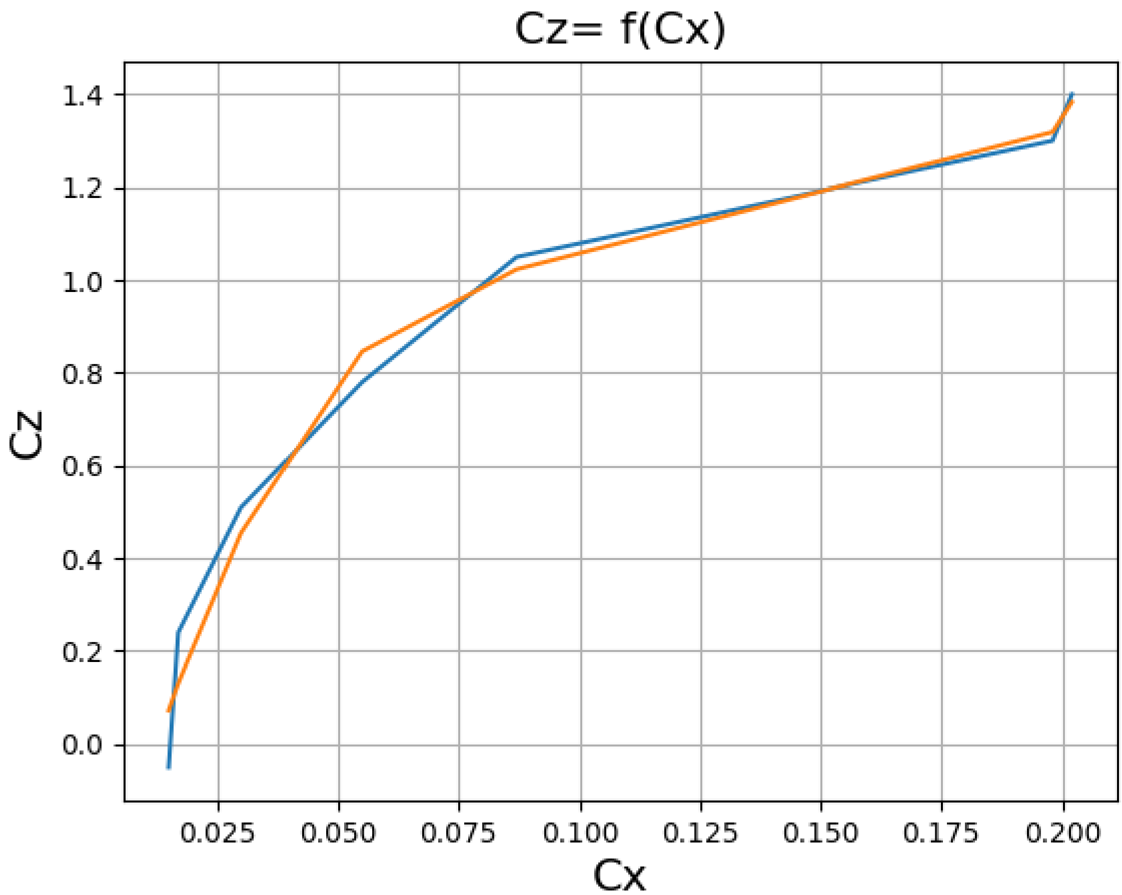
```
plt.grid(True)
plt.show()
```

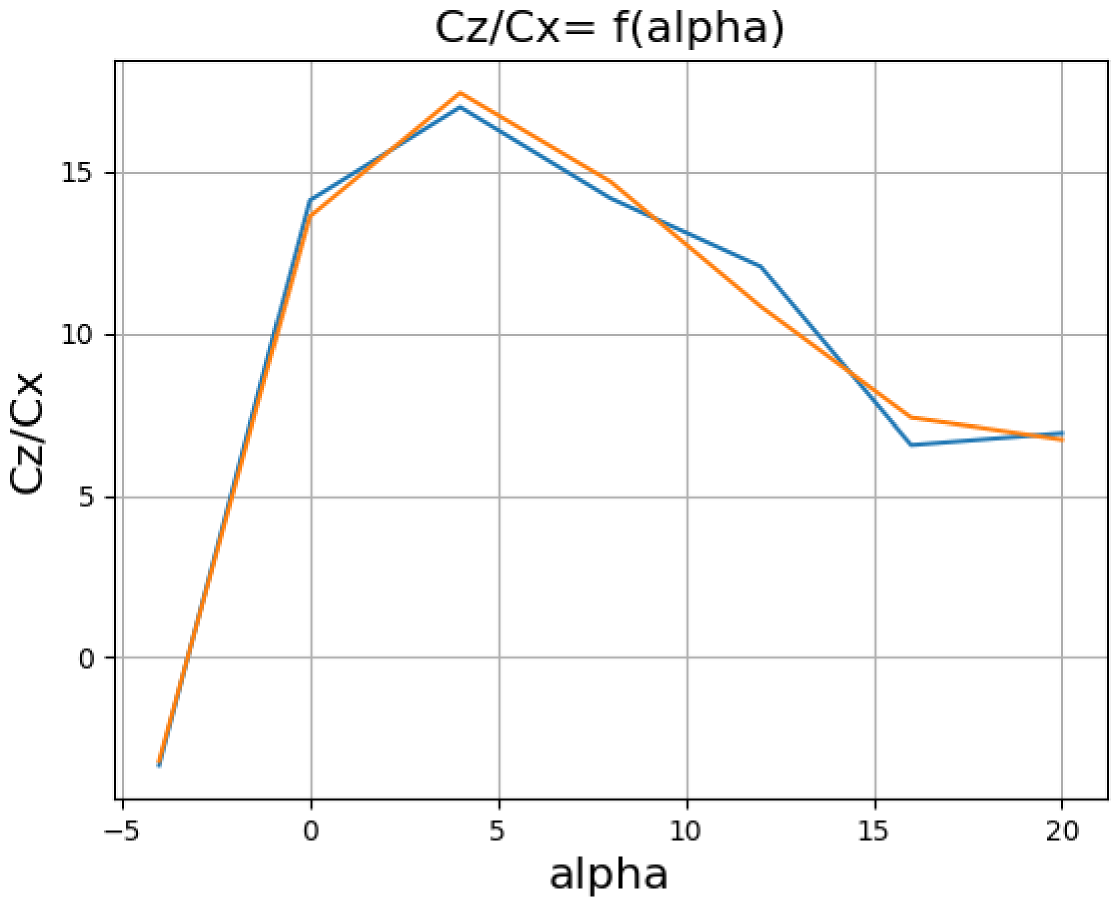
Result:

```
Python Console
 Cz/Cx(5)    :  [-3.33333 14.11765 17.       14.18182 12.06897  6.56566  6.93069]
 alpha[rad] :  [-0.06981  0.        0.06981  0.13963  0.20944  0.27925  0.34907]
 delta_alpha:  [ 0.0013  -0.00624 -0.01326 -0.02028 -0.0273  -0.0338  -0.0364 ]
 Cz^2        :  [0.0025 0.0576 0.2601 0.6084 1.1025 1.69   1.96  ]
 delta Cxi   :  [-0.0001 -0.0014 -0.0061 -0.0143 -0.0259 -0.0397 -0.0461]
 Cx(8)       :  [0.015, 0.017, 0.03, 0.055, 0.087, 0.198, 0.202]
 Cz/Cx(8)    :  [-3.33333 14.11765 17.       14.18182 12.06897  6.56566  6.93069]
```




Cx= f(alpha)
Cx
0.00
0.05
0.10
0.15
0.20
−5
0
5
10
15
20


Cz= f(Cx)
Cz
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200
Cx



Cz/Cx= f(alpha)
Cz/Cx
alpha
15
10
5
0
−5
0
5
10
15
20