Листинг программы расчета аэродинамических характеристик

Программа для определения коэффициента подъемной силы летательного аппарата

Аннотация

Программа реализована на языке Python с использованием библиотеки AeroBDSM для расчета аэродинамических характеристик. Выполняет расчет коэффициента подъемной силы с учетом интерференции между корпусом и несущими поверхностями, скоса потока, а также влияния числа Маха и угла атаки.

Основной код программы

```
import AeroBDSM
import math
import csv
import numpy as np
from tqdm import tqdm
import warnings
warnings.filterwarnings("ignore", category=RuntimeWarning)
class ProbabilityIntegral:
    def __init__(self):
        # Полная таблица значений   (x) с исправленными опечатками
        self.table = {
            # Первая часть таблицы (0.00 - 1.20)
            0.00: 0.0000, 0.01: 0.0080, 0.02: 0.0160, 0.03: 0.0239, 0.04: 0.0319,
            0.05: 0.0399, 0.06: 0.0478, 0.07: 0.0558, 0.08: 0.0638, 0.09: 0.0717,
            0.10: 0.0797, 0.11: 0.0876, 0.12: 0.0955, 0.13: 0.1034, 0.14: 0.1113,
            0.15: 0.1192, 0.16: 0.1271, 0.17: 0.1350, 0.18: 0.1428, 0.19: 0.1507,
            0.20: 0.1585, 0.21: 0.1663, 0.22: 0.1741, 0.23: 0.1819, 0.24: 0.1897,
            0.25: 0.1974, 0.26: 0.2051, 0.27: 0.2128, 0.28: 0.2205, 0.29: 0.2282,
            0.30: 0.2358, 0.31: 0.2434, 0.32: 0.2510, 0.33: 0.2586, 0.34: 0.2661,
            0.35: 0.2737, 0.36: 0.2812, 0.37: 0.2886, 0.38: 0.2961, 0.39: 0.3035,
            0.40: 0.3108, 0.41: 0.3182, 0.42: 0.3255, 0.43: 0.3328, 0.44: 0.3401,
            0.45: 0.3473, 0.46: 0.3545, 0.47: 0.3616, 0.48: 0.3688, 0.49: 0.3759,
            0.50: 0.3829, 0.51: 0.3899, 0.52: 0.3969, 0.53: 0.4039, 0.54: 0.4108,
            0.55: 0.4177, 0.56: 0.4245, 0.57: 0.4313, 0.58: 0.4381, 0.59: 0.4448,
            0.60: 0.4515, 0.61: 0.4581, 0.62: 0.4647, 0.63: 0.4713, 0.64: 0.4778,
            0.65: 0.4843, 0.66: 0.4907, 0.67: 0.4971, 0.68: 0.5035, 0.69: 0.5098,
            0.70: 0.5161, 0.71: 0.5223, 0.72: 0.5285, 0.73: 0.5346, 0.74: 0.5407,
            0.75: 0.5467, 0.76: 0.5527, 0.77: 0.5587, 0.78: 0.5646, 0.79: 0.5705,
            0.80: 0.5763, 0.81: 0.5821, 0.82: 0.5878, 0.83: 0.5935, 0.84: 0.5991,
            0.85: 0.6047, 0.86: 0.6102, 0.87: 0.6157, 0.88: 0.6211, 0.89: 0.6265,
            0.90: 0.6319, 0.91: 0.6372, 0.92: 0.6424, 0.93: 0.6476, 0.94: 0.6528,
            0.95: 0.6579, 0.96: 0.6629, 0.97: 0.6680, 0.98: 0.6729, 0.99: 0.6778,
            1.00: 0.6827, 1.01: 0.6875, 1.02: 0.6923, 1.03: 0.6970, 1.04: 0.7017,
            1.05: 0.7063, 1.06: 0.7109, 1.07: 0.7154, 1.08: 0.7199, 1.09: 0.7243,
            1.10: 0.7287, 1.11: 0.7330, 1.12: 0.7373, 1.13: 0.7415, 1.14: 0.7457,
            1.15: 0.7499, 1.16: 0.7540, 1.17: 0.7580, 1.18: 0.7620, 1.19: 0.7660,

            # Вторая часть таблицы (1.20 - 4.90)
            1.20: 0.7699, 1.21: 0.7737, 1.22: 0.7775, 1.23: 0.7813, 1.24: 0.7850,
            1.25: 0.7887, 1.26: 0.7923, 1.27: 0.7959, 1.28: 0.7995, 1.29: 0.8029,
            1.30: 0.8064, 1.31: 0.8098, 1.32: 0.8132, 1.33: 0.8165, 1.34: 0.8198,
            1.35: 0.8230, 1.36: 0.8262, 1.37: 0.8293, 1.38: 0.8324, 1.39: 0.8355,
            1.40: 0.8385, 1.41: 0.8415, 1.42: 0.8444, 1.43: 0.8473, 1.44: 0.8501,
            1.45: 0.8529, 1.46: 0.8557, 1.47: 0.8584, 1.48: 0.8611, 1.49: 0.8638,
            1.50: 0.8664, 1.51: 0.8690, 1.52: 0.8715, 1.53: 0.8740, 1.54: 0.8764,
            1.55: 0.8789, 1.56: 0.8812, 1.57: 0.8836, 1.58: 0.8859, 1.59: 0.8882,
            1.60: 0.8904, 1.61: 0.8926, 1.62: 0.8948, 1.63: 0.8969, 1.64: 0.8990,
            1.65: 0.9011, 1.66: 0.9031, 1.67: 0.9051, 1.68: 0.9070, 1.69: 0.9090,
            1.70: 0.9109, 1.71: 0.9127, 1.72: 0.9146, 1.73: 0.9164, 1.74: 0.9181,
            1.75: 0.9199, 1.76: 0.9216, 1.77: 0.9233, 1.78: 0.9249, 1.79: 0.9265,
            1.80: 0.9281, 1.81: 0.9297, 1.82: 0.9312, 1.83: 0.9328, 1.84: 0.9342,
            1.85: 0.9357, 1.86: 0.9371, 1.87: 0.9385, 1.88: 0.9399, 1.89: 0.9412,
            1.90: 0.9426, 1.91: 0.9439, 1.92: 0.9451, 1.93: 0.9464, 1.94: 0.9476,
            1.95: 0.9488, 1.96: 0.9500, 1.97: 0.9512, 1.98: 0.9523, 1.99: 0.9534,
            2.00: 0.9545, 2.05: 0.9596, 2.10: 0.9643, 2.15: 0.9684, 2.20: 0.9722,
            2.25: 0.9756, 2.30: 0.9786, 2.35: 0.9812, 2.40: 0.9836, 2.45: 0.9857,
            2.50: 0.9876, 2.55: 0.9892, 2.60: 0.9907, 2.65: 0.9920, 2.70: 0.9931,
            2.75: 0.9940, 2.80: 0.9949, 2.85: 0.9956, 2.90: 0.9963, 2.95: 0.9968,
```

```
            3.00: 0.99730, 3.10: 0.99806, 3.20: 0.99863, 3.30: 0.99903, 3.40: 0.99933,
            3.50: 0.99953, 3.60: 0.99968, 3.70: 0.99978, 3.80: 0.99986, 3.90: 0.99990,
            4.00: 0.99994, 4.417: 0.99999, 4.852: 0.999999, 5.327: 0.9999999
        }

        # Исправления опечаток из таблицы
        self.table[0.46] = 0.3545  # Исправлено с 0.3845
        self.table[0.49] = 0.3759  # Исправлено с 0.3769
        self.table[0.57] = 0.4313  # Исправлено с 0.4813
        self.table[0.68] = 0.5035  # Исправлено с 0.5095

    def get_value(self, x):
        """Получить значение   (x) с линейной интерполяцией"""
        # Если значение есть в таблице - возвращаем его
        if x in self.table:
            return self.table[x]

        # Получаем отсортированные ключи
        x_values = sorted(self.table.keys())

        # Если x меньше минимального значения
        if x < x_values[0]:
            return 0.0

        # Если x больше максимального значения
        if x > x_values[-1]:
            return 1.0  #   (x) стремится к 1 при x

        # Линейная интерполяция между ближайшими значениями
        for i in range(len(x_values) - 1):
            if x_values[i] <= x <= x_values[i + 1]:
                x1, x2 = x_values[i], x_values[i + 1]
                y1, y2 = self.table[x1], self.table[x2]
                return y1 + (y2 - y1) * (x - x1) / (x2 - x1)

        return self.table[x_values[-1]]

phi = ProbabilityIntegral()



# Глобальные переменные
l_f = 3.906  # длина фюзеляжа м
# M = 0.0
pi = math.pi
D = 0.178  # диаметр миделя, м aka диаметр миделевого сечения корпуса, м
D_I = 0.178  # диаметр корпуса в области передних консолей, м
D_II = 0.178  # диаметр корпуса в области задних консолей, м
D_bar = D / l_f  # относительный диаметр корпуса

l_nos = 0.47  # длина носа м
l_korm = 0.375  # длина кормы, м
S_m = (pi * pow(D, 2.0)) / 4.0  # площадь миделя , м^2
l_I = 0.484  # размах крыла м
l_II = 0.5808  # размах руля м

# Коэффициенты интерференции
k_aa_t = pow(1 + 0.41 * D_bar, 2)
K_aa_t = pow(1 + D_bar, 2)
eta_k = 1 # обратное суженеие

b_b = 0.2    #бортовая хорда

L_xv = 1.701  #длина хвостовой части(от конца бортовой хорды до конца короче)

c = 0.004 # толщина профиля
x_b = 1 #??????? что это ? стр.162
L_1 = x_b + ( b_b/ 2)
nu = 15.1*1e-6 #кин. кэф. вязкости воздуха стр.162
L1_bar=L_1/D
x_M = 1 #откуда??
y_v = 1.0

#для рулей:
eta_k_rl = 1
b_b_rl = 0.316
```

```
L_xv_rl = 1.701
L_1_rl = x_b + ( b_b_rl/ 2)
L1_bar_rl=L_1_rl/D
lambda_rl = 2.805
chi_05_rl = 0.420
zeta_rl = 0.0348
#////////////////////////////////////////////////////////
#для  get_psi_eps:
M_values = np.linspace(0.1,4.1,36)

alpha_p_values = np.linspace(0,0.436332,10) #от 0 до +25 градусов

phi_alpha_values = np.linspace(0, 0.436332, 10) #от 0 до +25 градусов

psi_I_values = np.linspace(0, 0.436332, 10) #от 0 до +25 градусов

psi_II_values = np.linspace(0, 0.436332, 10) #от 0 до +25 градусов

x_zI_II = 1
d_II = D
l_1c_II = 0.231
zeta_II  = zeta_rl
b_b_II = 0.316
chi_0_II =0.420

#////////////////////////////////////////////


lambda_kr = 2.49  # удлинение несущей поверхности
chi_05_kr = 0.510  # угол стреловидности по линии середин хорд, рад
bar_c_kr = 0.224  # относительная толщина профиля
zeta_kr = 0.2  # обратное сужение несущей поверхности
# Дополнительные параметры
b_kr = 0.1  # Хорда крыла, м
b_op = 0.025  # Хорда оперения, м
l_raszmah_kr = 0.498  # Размах крыла, м
l_raszmah_rl = 0.651  # Размах оперения, м

# ////////////////////////////////////////////
# расчет c^\alpha_y

S_f = 2.1842

S_kr = 0.0996

S_rl = 0.15107

S = S_f + S_kr + S_rl

S_f_bar = S_f / S  # Относительная площадь фюзеляжа
S_1_bar = S_kr / S  # Относительная площадь оперения
S_2_bar = S_rl / S  # Относительная площадь крыла



# ////////////////////////////////////////////

# Площади поверхностей
S_Kons = 4 * 0.03375 * 0.008  # Площадь консолей крыла, м
S_op = 2 * 0.033 * 0.025  # Площадь оперения, м

L_A = 1
b_A = 1.5


# # Расчетные параметры крыльевых поверхностей
# l_raszmah_kons = l_raszmah - D  # Размах консоли крыла
# lamb = pow(l_raszmah_kons, 2.0) / S_Kons  # Удлинение крыла
# lamb_op = pow(l_raszmah_rl, 2.0) / (S_op + pow(b_kr, 2.0))  # Удлинение оперения

# Итоговые параметры
S_f = 2.0 * pi * D * (l_f - D / 2.0) + 2.0 * pi * pow(D / 2.0, 2.0)  # Площадь поверхности фюз
    еляжа, м

def save_data_to_csv(x, y, filename):
    """Сохраняет данные в CSV файл"""
```

```
    with open(filename, 'w', newline='') as file:
        writer = csv.writer(file)
        writer.writerow(['x', 'y'])
        for i in range(len(x)):
            writer.writerow([x[i], y[i]])

def main():
    angles_deg = [-10, -5, 0, 5, 10]
    lambda_Nos = l_nos / D
    lambda_Cil = l_f / D

    # Первый расчет - носовая часть
    with open('all_angles_data.csv', 'w', newline='') as file1:
        writer1 = csv.writer(file1)
        header1 = ['Mach'] + [f'alpha_{angle}' for angle in angles_deg]
        writer1.writerow(header1)

        M = 0.5
        while M <= 4.1:
            result = AeroBDSM.get_c_y_alpha_NosCil_Con(M, lambda_Nos, lambda_Cil)
            if result.ErrorCode == 0:
                c_y_alpha = result.Value
                row = [M]
                for angle_deg in angles_deg:
                    angle_rad = angle_deg * math.pi / 180.0
                    c_y = c_y_alpha * angle_rad
                    row.append(c_y)
                writer1.writerow(row)
            M += 0.1

    # Второй расчет - крыло изолированное
    with open('krylo_isP.csv', 'w', newline='') as file2:
        writer2 = csv.writer(file2)
        header2 = ['Mach'] + [f'alpha_{angle}' for angle in angles_deg]
        writer2.writerow(header2)

        M = 0.5
        while M <= 4.1:
            result2 = AeroBDSM.get_c_y_alpha_IsP(M, lambda_kr, bar_c_kr, chi_05_kr, zeta_kr)
            if result2.ErrorCode == 0:
                c_y_alpha = result2.Value
                row = [M]
                for angle_deg in angles_deg:
                    angle_rad = angle_deg * math.pi / 180.0
                    c_y = c_y_alpha * angle_rad
                    row.append(c_y)
                writer2.writerow(row)
            M += 0.1

    with open('Kappa_aa_kr.csv', 'w', newline='') as file4:
        writer4 = csv.writer(file4)
        header4 = ['Mach', 'K_aa'] + [f'alpha_{angle}' for angle in angles_deg]
        writer4.writerow(header4)

        M = 0.5
        while M <= 4.1:
            V = 342 * M

            # Исправление для избежания math domain error
            if M < 1:
                sqrt_term = 0  # Для M < 1 используем 0
            else:
                sqrt_term = math.sqrt(M**2 - 1)

            b_b_bar = b_b / ((math.pi/2)*D* sqrt_term) if sqrt_term > 0 else 0
            if b_b_bar < 0 or M ==2.0000000000000004  :
                print(f'b_b_bar = {b_b_bar:.3f}')

            L_xv_bar = (L_xv/ ((math.pi/2)*D* sqrt_term)) if sqrt_term > 0 else 0
            if L_xv_bar < 0 or M == 2.0000000000000004 :
                print(f'L_xv_bar = {L_xv_bar:.3f}')

            # Проверка деления на ноль для F_L_xv
            if b_b_bar > 0 and c > 0:
                sqrt_2c = math.sqrt(2*c)
```

```
285                 F_L_xv = 1 - (math.sqrt(math.pi) / (2 * b_b_bar * sqrt_2c)) *
                        (phi.get_value((b_b_bar + L_xv_bar)* sqrt_2c) - phi.get_value(L_xv_bar *
                        sqrt_2c))
286             else:
287                 F_L_xv = 1.0
288
289             if F_L_xv < 0 or M == 2.00000000000000042:
290                 print(f'F_L_xv = {F_L_xv:.3f}')
291
292             delta_zvz_bar_kr = (0.093 / ((V * L_1) / nu)**(1/5)) * (L_1 / D) * (1 + 0.4*M +
                    0.147 * M**2 - 0.006 * M**3)
293             if delta_zvz_bar_kr < 0 or M == 2.0000000000000004:
294                 print(f'delta_zvz_bar_kr = {delta_zvz_bar_kr:.3f}')
295
296             x_ps = (1 - ((2 * D_bar)/(1 - D_bar**2)) * delta_zvz_bar_kr) * (1 - ((D_bar *
                    (eta_k-1))/(1 - D_bar) * (eta_k + 1)) * delta_zvz_bar_kr)
297             if x_ps < 0 or M == 2.0000000000000004 :
298                 print(f'x_ps = {x_ps:.3f}')
299
300             x_nos = 0.6 + 0.4 * (1 - math.exp(-0.5 * L1_bar))
301             if x_nos < 0 or M == 2.0000000000000004:
302                 print(f'x_nos = {x_nos:.3f}')
303
304             k_aa_zvz = K_aa_t * ((1 + 3*D_bar - (1 / eta_k) * D_bar * (1 - D_bar)) / (1 +
                    D_bar)**2)
305             if k_aa_zvz < 0 or M == 2.00000000000000042:
306                 print(f'k_aa_zvz = {k_aa_zvz:.3f}')
307
308             K_aa_zvz = 1 + 3 * D_bar - ((D_bar * (1 - D_bar)) / eta_k)
309             if K_aa_zvz < 0 or M == 2.0000000000000004:
310                 print(f'K_aa_zvz = {K_aa_zvz:.3f}')
311
312             k_aa =  K_aa_zvz * x_ps * x_M * x_nos
313             if M >=1 :
314                 K_aa = (k_aa_zvz + (K_aa_zvz - k_aa_zvz) * F_L_xv) * x_ps * x_M * x_nos
315             else:
316                 K_aa = K_aa_zvz * F_L_xv * x_ps * x_M * x_nos
317
318             row = [M, K_aa]
319             for angle_deg in angles_deg:
320                 angle_rad = angle_deg * math.pi / 180.0
321                 row.append(K_aa)
322
323             writer4.writerow(row)
324
325             M += 0.1
326     # пятый расчет крыло + good интерфер-я
327     with open('krylo_isP_Intrf.csv', 'w', newline='') as file5:
328         writer5 = csv.writer(file5)
329         header5 = ['Mach'] + [f'alpha_{angle}' for angle in angles_deg]
330         writer5.writerow(header5)
331
332         M = 0.5
333         while M <= 4.1:
334             result5 = AeroBDSM.get_c_y_alpha_IsP(M, lambda_kr, bar_c_kr, chi_05_kr, zeta_kr)
335             if result5.ErrorCode == 0:
336                 c_y_alpha = result5.Value * K_aa
337                 row = [M]
338                 for angle_deg in angles_deg:
339                     angle_rad = angle_deg * math.pi / 180.0
340                     c_y = c_y_alpha * angle_rad
341                     row.append(c_y)
342                 writer5.writerow(row)
343             M += 0.1
344
345
346     # шестой расчет - руль изолированный
347     with open('rul_isP.csv', 'w', newline='') as file6:
348         writer6 = csv.writer(file6)
349         header6 = ['Mach'] + [f'alpha_{angle}' for angle in angles_deg]
350         writer6.writerow(header6)
351
352         M = 0.5
353         while M <= 4.1:
354             result6 = AeroBDSM.get_c_y_alpha_IsP(M, lambda_rl, bar_c_kr, chi_05_rl, zeta_rl)
355             if result6.ErrorCode == 0:
```

```
                c_y_alpha = result6.Value
                row = [M]
                for angle_deg in angles_deg:
                    angle_rad = angle_deg * math.pi / 180.0
                    c_y = c_y_alpha * angle_rad
                    row.append(c_y)
                writer6.writerow(row)
            M += 0.1
    print(60*'=')
    with open('Kappa_aa_rl.csv', 'w', newline='') as file7:
        writer7 = csv.writer(file7)
        header7 = ['Mach', 'K_aa'] + [f'alpha_{angle}' for angle in angles_deg]
        writer7.writerow(header7)

        M = 0.5
        while M <= 4.1:
            V = 342 * M

            # Исправление для избежания math domain error
            if M < 1:
                sqrt_term = 0  # Для M < 1 используем 0
            else:
                sqrt_term = math.sqrt(M**2 - 1)

            b_b_bar_rl = b_b / ((math.pi/2)*D* sqrt_term) if sqrt_term > 0 else 0
            if b_b_bar_rl < 0 or M ==2.0000000000000004  :
                print(f'b_b_bar_rl = {b_b_bar_rl:.3f}')

            L_xv_bar_rl = (L_xv_rl/ ((math.pi/2)*D* sqrt_term)) if sqrt_term > 0 else 0
            if L_xv_bar_rl < 0 or M == 2.0000000000000004 :
                print(f'L_xv_bar_rl = {L_xv_bar_rl:.3f}')

            # Проверка деления на ноль для F_L_xv
            if b_b_bar_rl > 0 and c > 0:
                sqrt_2c = math.sqrt(2*c)
                F_L_xv_rl = 1 - (math.sqrt(math.pi) / (2 * b_b_bar_rl * sqrt_2c)) *
                    (phi.get_value((b_b_bar_rl + L_xv_bar_rl)* sqrt_2c) -
                    phi.get_value(L_xv_bar_rl * sqrt_2c))
            else:
                F_L_xv_rl = 1.0

            if F_L_xv_rl < 0 or M == 2.00000000000000042:
                print(f'F_L_xv_rl = {F_L_xv_rl:.3f}')

            delta_zvz_bar_rl = (0.093 / ((V * L_1_rl) / nu)**(1/5)) * (L_1_rl / D) * (1 +
                0.4*M + 0.147 * M**2 - 0.006 * M**3)
            if delta_zvz_bar_rl < 0 or M == 2.0000000000000004:
                print(f'delta_zvz_bar_rl = {delta_zvz_bar_rl:.3f}')

            x_ps_rl = (1 - ((2 * D_bar)/(1 - D_bar**2)) * delta_zvz_bar_rl) * (1 - ((D_bar *
                (eta_k_rl-1))/(1 - D_bar) * (eta_k_rl + 1)) * delta_zvz_bar_rl)
            if x_ps_rl < 0 or M == 2.0000000000000004 :
                print(f'x_ps_rl = {x_ps_rl:.3f}')

            x_nos_rl = 0.6 + 0.4 * (1 - math.exp(-0.5 * L1_bar_rl))
            if x_nos_rl < 0 or M == 2.0000000000000004:
                print(f'x_nos_rl = {x_nos_rl:.3f}')

            k_aa_zvz_rl = K_aa_t * ((1 + 3*D_bar - (1 / eta_k_rl) * D_bar * (1 - D_bar)) / (1
                + D_bar)**2)
            if k_aa_zvz_rl < 0 or M == 2.00000000000000042:
                print(f'k_aa_zvz_rl = {k_aa_zvz_rl:.3f}')

            K_aa_zvz_rl = 1 + 3 * D_bar - ((D_bar * (1 - D_bar)) / eta_k_rl)
            if K_aa_zvz_rl < 0 or M == 2.0000000000000004:
                print(f'K_aa_zvz_rl = {K_aa_zvz_rl:.3f}')

            if M >=1 :
                K_aa_rl = (k_aa_zvz_rl + (K_aa_zvz_rl - k_aa_zvz_rl) * F_L_xv_rl) * x_ps_rl *
                    x_M * x_nos_rl
            else:
                K_aa_rl = K_aa_zvz_rl * F_L_xv_rl * x_ps_rl * x_M* x_nos_rl

            row = [M, K_aa]
            for angle_deg in angles_deg:
                angle_rad = angle_deg * math.pi / 180.0
```

```
                row.append(K_aa_rl)

            writer7.writerow(row)

            M += 0.1

            #Учет скоса потока
    with open('z_b_v.csv', 'w', newline='') as file8:
        writer8 = csv.writer(file8)
        header8 = ['Mach', 'z_b']
        writer8.writerow(header8)
        M = 0.5
        while M <= 4.1:
            result8 = AeroBDSM.get_bar_z_v(M, lambda_kr, chi_05_kr, zeta_kr)
            if result8.ErrorCode == 0:
                z_b = result8.Value
                row = [M, z_b]
                writer8.writerow(row)
            M += 0.1


    # Учет скоса потока - расчет i_v
    with open('i_v.csv', 'w', newline='') as file9:
        writer9 = csv.writer(file9)
        header9 = ['Mach', 'i_v']
        writer9.writerow(header9)
        M = 0.5
        while M <= 4.1:
            # Сначала получаем z_b для текущего M
            result8 = AeroBDSM.get_bar_z_v(M, lambda_kr, chi_05_kr, zeta_kr)
            if result8.ErrorCode == 0:
                z_b = result8.Value
                # Затем рассчитываем i_v с полученным z_b
                result9 = AeroBDSM.get_i_v(zeta_rl, D, l_raszmah_rl, y_v, z_b)
                if result9.ErrorCode == 0:
                    i_v = result9.Value
                    row = [M, i_v]
                    writer9.writerow(row)
            M += 0.1

    with open('psi_eps.csv', 'w', newline='') as file10:
        writer10 = csv.writer(file10)
        header10 = ['Mach', 'alpha_p','phi_alpha','psi_I','psi_II','z_v', 'psi_eps']
        writer10.writerow(header10)
        total_iterations = len(psi_II_values) * len(psi_I_values) * len(phi_alpha_values) *
            len(alpha_p_values) * len(M_values)
        with tqdm(total=total_iterations, desc="Расчет psi_eps") as pbar:
            for psi_II in psi_II_values:
                for psi_I in psi_I_values:
                    for phi_alpha in phi_alpha_values:
                        for alpha_p in alpha_p_values:
                            for M in M_values:
                                result8 = AeroBDSM.get_bar_z_v(M, lambda_kr, chi_05_kr,
                                    zeta_kr)
                                if result8.ErrorCode == 0:
                                    z_v = result8.Value
                                    result10 = AeroBDSM.get_psi_eps(M, alpha_p, phi_alpha,
                                        psi_I, psi_II, z_v, y_v, x_zI_II, d_II,
                                        l_1c_II, zeta_II, b_b_II, chi_0_II)
                                    if result10.ErrorCode == 0:
                                        psi_eps = result10.Value
                                        row = [M, alpha_p, phi_alpha, psi_I, psi_II, z_v,
                                            psi_eps]
                                        writer10.writerow(row)
                                pbar.update(1)

    with open('eps_alpha_sr.csv', 'w', newline='') as file11:
        writer11 = csv.writer(file11)
        header11 = ['eps_alpha_sr', 'Mach', 'alpha_p','phi_alpha','psi_I','psi_II','z_v',
            'psi_eps', 'i_v', 'k_aa', 'K_aa_rl']
        writer11.writerow(header11)
        total_iterations = len(psi_II_values) * len(psi_I_values) * len(phi_alpha_values) *
            len(alpha_p_values) * len(M_values)
        with tqdm(total=total_iterations, desc="eps_alpha_sr") as pbar:
            for psi_II in psi_II_values:
                for psi_I in psi_I_values:
```

```
                    for phi_alpha in phi_alpha_values:
                        for alpha_p in alpha_p_values:
                            for M in M_values:
                                result8 = AeroBDSM.get_bar_z_v(M, lambda_kr, chi_05_kr,
                                    zeta_kr)
                                if result8.ErrorCode == 0:
                                    z_v = result8.Value
                                    result10 = AeroBDSM.get_psi_eps(M, alpha_p, phi_alpha,
                                        psi_I, psi_II, z_v, y_v, x_zI_II, d_II,
                                        l_1c_II, zeta_II, b_b_II, chi_0_II)

                                    if result10.ErrorCode == 0:

                                        result9 = AeroBDSM.get_i_v(zeta_rl, D, l_raszmah_rl,
                                            y_v, z_v)
                                        i_v = result9.Value

                                        psi_eps = result10.Value

                                        result2 = AeroBDSM.get_c_y_alpha_IsP(M, lambda_kr,
                                            bar_c_kr, chi_05_kr, zeta_kr)

                                        c_y_alpha_iz_kr = result2.Value

                                        V = 342 * M
                                        # Исправление для избежания math domain error
                                        if M < 1:
                                            sqrt_term = 0  # Для M < 1 используем 0
                                        else:
                                            sqrt_term = math.sqrt(M**2 - 1)

                                        b_b_bar_rl = b_b / ((math.pi/2)*D* sqrt_term) if
                                            sqrt_term > 0 else 0

                                        L_xv_bar_rl = (L_xv_rl/ ((math.pi/2)*D* sqrt_term))
                                            if sqrt_term > 0 else 0


                                        # Проверка деления на ноль для F_L_xv
                                        if b_b_bar_rl > 0 and c > 0:
                                            sqrt_2c = math.sqrt(2*c)
                                            F_L_xv_rl = 1 - (math.sqrt(math.pi) / (2 *
                                                b_b_bar_rl * sqrt_2c)) *
                                                (phi.get_value((b_b_bar_rl + L_xv_bar)*
                                                sqrt_2c) - phi.get_value(L_xv_bar * sqrt_2c))
                                        else:
                                            F_L_xv_rl = 1.0

                                        delta_zvz_bar_rl = (0.093 / ((V * L_1_rl) /
                                            nu)**(1/5)) * (L_1_rl / D) * (1 + 0.4*M + 0.147 *
                                            M**2 - 0.006 * M**3)

                                        x_ps_rl = (1 - ((2 * D_bar)/(1 - D_bar**2)) *
                                            delta_zvz_bar_rl) * (1 - ((D_bar * (eta_k-1))/(1
                                            - D_bar) * (eta_k_rl + 1)) * delta_zvz_bar_rl)

                                        x_nos_rl = 0.6 + 0.4 * (1 - math.exp(-0.5 *
                                            L1_bar_rl))


                                        k_aa_zvz_rl = K_aa_t * ((1 + 3*D_bar - (1 / eta_k) *
                                            D_bar * (1 - D_bar)) / (1 + D_bar)**2)

                                        K_aa_zvz_rl = 1 + 3 * D_bar - ((D_bar * (1 - D_bar))
                                            / eta_k_rl)

                                        if M >=1 :
                                            K_aa_rl = (k_aa_zvz_rl + (K_aa_zvz_rl -
                                                k_aa_zvz_rl) * F_L_xv_rl) * x_ps_rl * x_M *
                                                x_nos_rl
                                        else:
                                            K_aa_rl = K_aa_zvz_rl * F_L_xv_rl * x_ps_rl * x_M
                                                * x_nos_rl

                                            #////////////////////////////////////////////////////////////
```

```
                                    if M < 1:
                                        sqrt_term = 0  # Для M < 1 используем 0
                                    else:
                                        sqrt_term = math.sqrt(M**2 - 1)

                                    b_b_bar = b_b / ((math.pi/2)*D* sqrt_term) if
                                        sqrt_term > 0 else 0

                                    L_xv_bar = (L_xv/ ((math.pi/2)*D* sqrt_term)) if
                                        sqrt_term > 0 else 0

                                    # Проверка деления на ноль для F_L_xv
                                    if b_b_bar > 0 and c > 0:
                                        sqrt_2c = math.sqrt(2*c)
                                        F_L_xv = 1 - (math.sqrt(math.pi) / (2 * b_b_bar *
                                            sqrt_2c)) * (phi.get_value((b_b_bar +
                                            L_xv_bar)* sqrt_2c) - phi.get_value(L_xv_bar
                                            * sqrt_2c))
                                    else:
                                        F_L_xv = 1.0

                                    delta_zvz_bar_kr = (0.093 / ((V * L_1) / nu)**(1/5))
                                        * (L_1 / D) * (1 + 0.4*M + 0.147 * M**2 - 0.006 *
                                        M**3)

                                    x_ps = (1 - ((2 * D_bar)/(1 - D_bar**2)) *
                                        delta_zvz_bar_kr) * (1 - ((D_bar * (eta_k-1))/(1
                                        - D_bar) * (eta_k + 1)) * delta_zvz_bar_kr)

                                    x_nos = 0.6 + 0.4 * (1 - math.exp(-0.5 * L1_bar))

                                    k_aa_zvz = K_aa_t * ((1 + 3*D_bar - (1 / eta_k) *
                                        D_bar * (1 - D_bar)) / (1 + D_bar)**2)

                                    K_aa_zvz = 1 + 3 * D_bar - ((D_bar * (1 - D_bar)) /
                                        eta_k)

                                    k_aa =  K_aa_zvz * x_ps * x_M * x_nos

                                    eps_alpha_sr = (57.3/(2*math.pi)) * (i_v/z_v) *
                                        (l_raszmah_kr/l_raszmah_rl) *
                                        (c_y_alpha_iz_kr/lambda_kr) * (k_aa/K_aa_rl) *
                                        psi_eps
                                    row = [eps_alpha_sr, M, alpha_p, phi_alpha, psi_I,
                                        psi_II, z_v, psi_eps, i_v, k_aa, K_aa_rl ]
                                    writer11.writerow(row)
                                pbar.update(1)



with open('kappa_q_nos.csv', 'w', newline='') as file12:
    writer12 = csv.writer(file12)
    header12 = ['Mach', 'kappa_q']
    writer12.writerow(header12)
    M = 1
    while M <= 4.1:
        result12 = AeroBDSM. get_kappa_q_Nos_Con(M, lambda_Nos)
        if result12.ErrorCode == 0:
            kappa_q = result12.Value
            row = [M, kappa_q]
            writer12.writerow(row)
        M += 0.1


with open('kappa_q.csv', 'w', newline='') as file13:
    writer13 = csv.writer(file13)
    header13 = ['Mach', 'kappa_q']
    writer13.writerow(header13)
    M = 0.5
    while M <= 4.1:
        result13 = AeroBDSM.  get_kappa_q_IsP(M, L_A, b_A)
        if result13.ErrorCode == 0:
            kappa_q = result13.Value
            row = [M, kappa_q]
            writer13.writerow(row)
```

```
            M += 0.1

    #расчет c_^\alpha_y
    with open('c_y_alpha_sum.csv', 'w', newline='') as file14:
        writer14 = csv.writer(file14)
        header14 = ['Mach'] + [f'alpha_{angle}' for angle in angles_deg]
        writer14.writerow(header14)

        M = 0.5
        while M <= 4.1:
            # Получаем все необходимые значения для текущего M
            result = AeroBDSM.get_c_y_alpha_NosCil_Con(M, lambda_Nos, lambda_Cil)
            result2 = AeroBDSM.get_c_y_alpha_IsP(M, lambda_kr, bar_c_kr, chi_05_kr, zeta_kr)
            result6 = AeroBDSM.get_c_y_alpha_IsP(M, lambda_rl, bar_c_kr, chi_05_rl, zeta_rl)
            result12 = AeroBDSM.get_kappa_q_Nos_Con(M, lambda_Nos)
            result13 = AeroBDSM.get_kappa_q_IsP(M, L_A, b_A)

            # Проверяем все ошибки
            if (result.ErrorCode == 0 and result2.ErrorCode == 0 and
                result6.ErrorCode == 0 and result12.ErrorCode == 0 and
                result13.ErrorCode == 0):

                # Вычисляем компоненты
                c_ya_f = result.Value
                c_ya_1 = result2.Value * K_aa
                c_ya_2 = result6.Value * K_aa_rl

                # Суммарный коэффициент для разных углов атаки
                row = [M]
                for angle_deg in angles_deg:
                    angle_rad = angle_deg * math.pi / 180.0
                    c_ya_1_sum = (c_ya_f * S_f_bar +
                                  c_ya_1 * S_1_bar * result12.Value +
                                  c_ya_2 * S_2_bar * result13.Value) * angle_rad
                    row.append(c_ya_1_sum)

                writer14.writerow(row)
            M += 0.1


if __name__ == "__main__":
    main()
print(f'Расчет успешно завершен')
```

Листинг 1: Полный код программы расчета (main.py)

Описание основных модулей

Класс ProbabilityIntegral

Реализует таблицу значений интеграла вероятностей с линейной интерполяцией для промежуточных значений.

Глобальные параметры

- Геометрические характеристики летательного аппарата
- Параметры крыла и рулевых поверхностей
- Коэффициенты интерференции
- Диапазоны чисел Маха и углов атаки для расчетов

Функция main()

Выполняет последовательный расчет:

1. Носовой части летательного аппарата
2. Изолированного крыла
3. Коэффициентов интерференции

4. Скоса потока
5. Суммарного коэффициента подъемной силы