Задания заключительного этапа **(полуфинала)**
Всероссийской олимпиады студентов «Я – профессионал»
по направлению «**Разработка беспилотных воздушных судов**»

Категория участия «Бакалавриат»

Вариант № 1

Задача № 1

Рассчитайте размах крыла, длины корневой и концевой хорд трапециевидного крыла БЛА самолётного типа, если площадь крыла 10 м^2, сужение 2, удлинение 7. Округлите ответ до десятых.

Решение (основные вычисления):

$$\text{Размах} = \sqrt{\text{Удлинение} * \text{Площадь}} = \sqrt{7 * 10\text{м}^2} = 8{,}4 \text{ м}$$

Трапециевидное крыло ⋙

$$\text{Площадь} = \text{размах} \left(\frac{\text{корневая хорда} + \text{концевая хорда}}{2}\right)$$

$$\text{Концевая хорда} = \frac{\text{корневая хорда}}{\text{сужение}}$$

$$\text{Корневая хорда} = \frac{2 * \text{сужение} * \text{площадь}}{\text{размах} * (1 + \text{сужение})} = \frac{2 * 2 * 10\text{м}^2}{8{,}4\text{м} * (1 + 2)} = 1{,}6 \text{ м}$$

$$\text{Концевая хорда} = \frac{\text{корневая хорда}}{\text{сужение}} = \frac{1{,}6}{2} = 0{,}8 \text{ м}$$

Ответ: размах крыла 8,4 м; корневая хорда 1,6 м; концевая хорда 0,8 м.

Задача № 2

БЛА самолётного типа снабжен гидросистемой. Дроссель клапана гидросистемы индуктивностью L = 100 мГн и внутренним сопротивлением R = 10 Ом подключили к источнику постоянного напряжения 28 В. Какова будет величина тока в цепи через 10 мс после подключения дросселя? Ответ дать с точностью до сотых.

Решение (основные вычисления):

Необходимо вывести формулу. Сила тока для нашего случая изменяется по закону:

$$I = I_0\left(1 - e^{-\frac{t}{\tau}}\right), \text{где } \tau = \frac{L}{R},\ I_0 = \frac{E}{R}$$

Тогда получим:

$$I = \frac{E}{R}\left(1 - e^{-\frac{R \cdot t}{L}}\right)$$

$$I = \frac{28}{10}\left(1 - e^{-\frac{10 \cdot 0{,}01}{100 \cdot 10^{-3}}}\right) = 2{,}8\left(1 - \frac{1}{e^1}\right) = 1{,}77 \text{ А} = 1769{,}94 \text{ мА}$$

Ответ: 1,77 А.

Задача № 3

Управляемый вектор тяги беспилотного воздушного судна (БВС) схемы тандем (далее – конвертоплан) (рисунок 1) позволяет совершать прямолинейный горизонтальный полёт, сбалансированный при помощи аэродинамических сил и направления тяги винтомоторной группы (ВМГ).

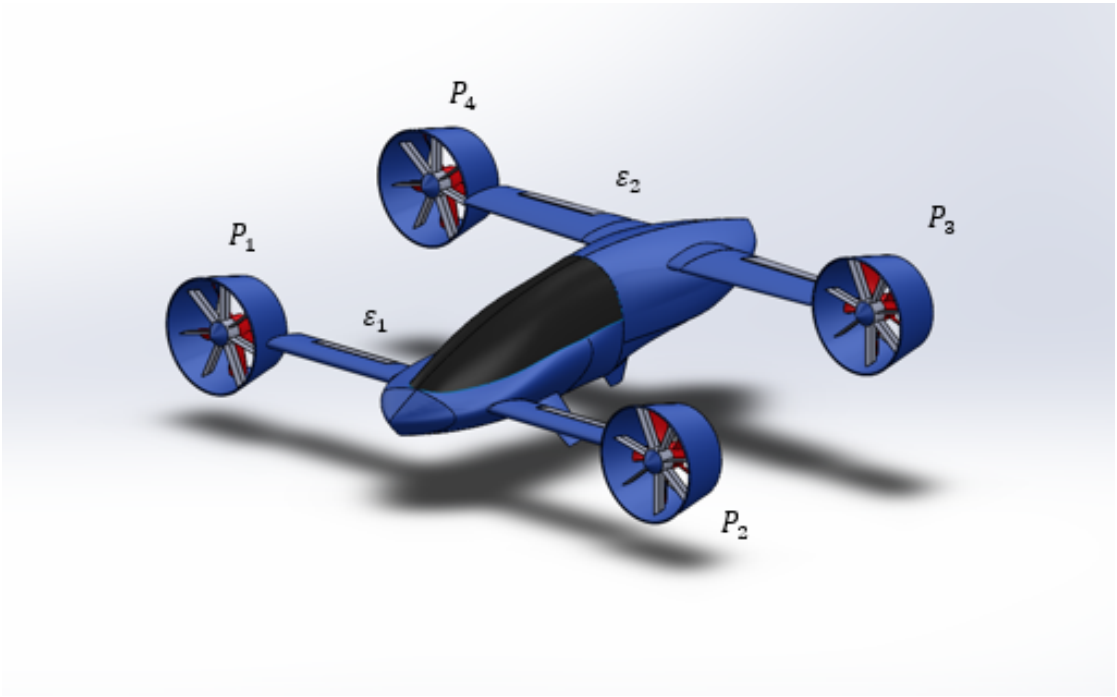


Рисунок 1. Внешний облик БВС, конвертоплан схемы тандем.

Угловые положения осей вращения двигателей (сил тяги $\vec{P}_i$) определяются углами поворота этих осей вокруг поворотных осей двигателей $O_{д1}Z_{д1}$, $O_{д2}Z_{д2}$ на угол ε_1 для первой пары и вокруг осей $O_{д3}Z_{д3}$, $O_{д4}Z_{д4}$ на угол ε_2 для второй пары соответственно.

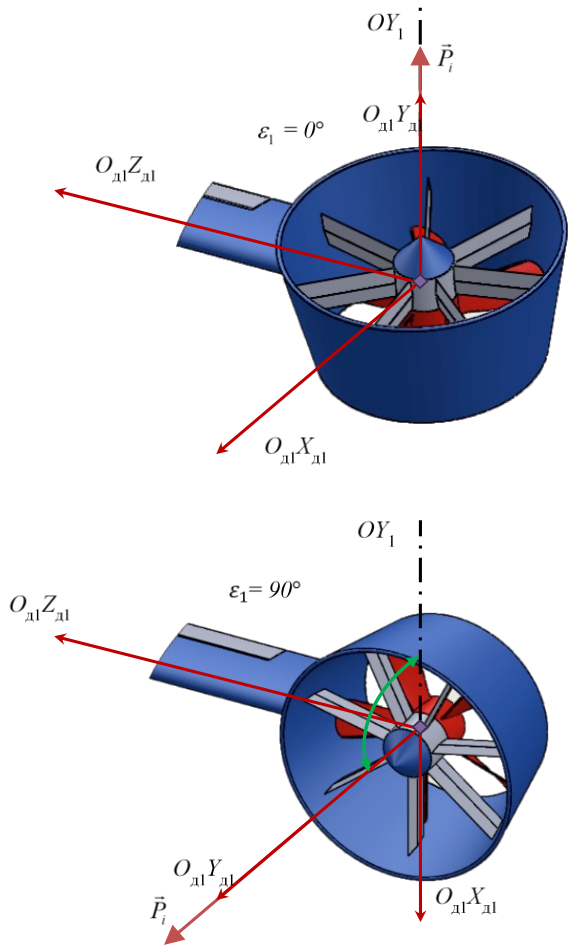


Рисунок 2. Вращение ВМГ вокруг оси ε_j.

В установившимся режиме полёта сумма моментов, действующих на БВС, равна нулю, и он находится в состоянии балансировки – для дальнейших расчётов необходимо воспользоваться уравнениями его установившегося движения. Схема сил, действующих на БВС в самолётном режиме, приведена на рисунке 3.

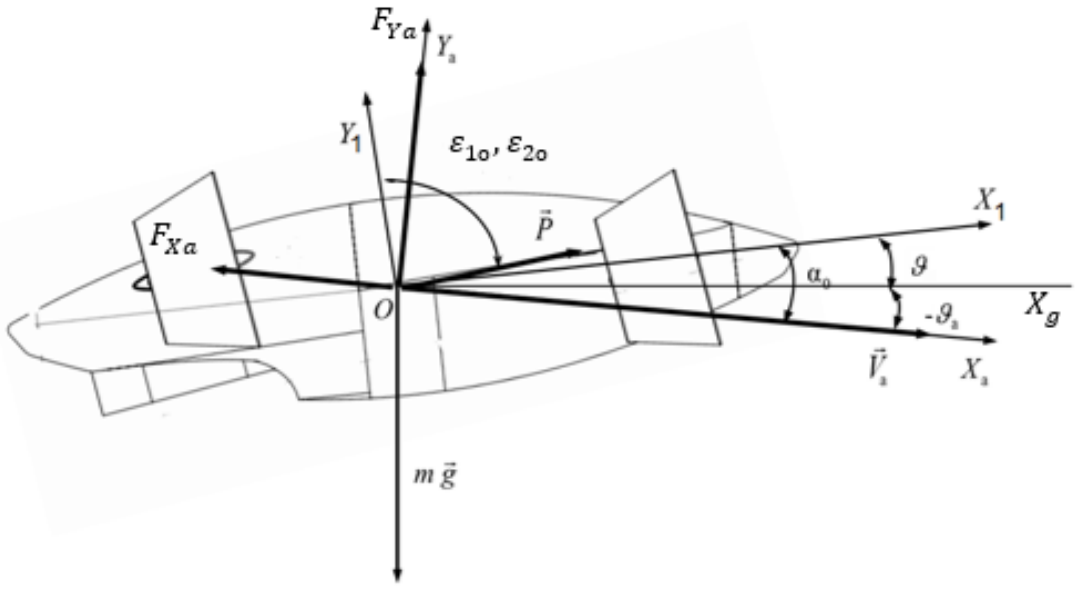


Рисунок 3. Силы, действующие на БВС в самолётном режиме.

Сила лобового сопротивления F_{Xa} направлена противоположно вектору воздушной скорости V_a БВС и равна:

$$F_{Xa} = c_x(\alpha)\frac{\rho V^2}{2}S.$$

Подъёмная сила F_{Ya} направлена перпендикулярно вектору воздушной скорости V_a БВС и равна:

$$F_{Ya} = c_y(\alpha)\frac{\rho V^2}{2}S.$$

Вектор силы тяжести направлен от центра масс БВС перпендикулярно вниз:

$$G = m \cdot g.$$

Конвертоплан совершает установившийся горизонтальный прямолинейный полёт (V = const, $\theta = 0$, H = const). Определите величину углов отклонения оси винтомоторной группы $\varepsilon_1, \varepsilon_2$, а также сумму тяг ВМГ $\sum_{i=1}^{4} P$, при условии, что $\varepsilon_1 = \varepsilon_2$. При решении данной задачи вращением БВС вокруг центра масс пренебречь. Ответ указать в градусах и Н с точностью до сотых.

При расчётах принять следующие параметры:

- ускорение свободного падения $g = 9{,}806$ м/с^2;
- плотность воздушной среды на текущей высоте $\rho = 1{,}2135$ кг/м^3;
- масса БВС $m = 108$ кг;
- характерная площадь крыла $S = 2{,}18$ м^2;
- истинная скорость $V_a = 40$ м/с;
- угол наклона траектории $\theta = 0°$;
- угол атаки $\alpha = 4{,}2°$;
- коэффициент лобового сопротивления $c_x(\alpha) = 0{,}0825$;
- коэффициент подъемной силы $c_y(\alpha) = 0{,}31167$.

Решение (основные вычисления):

Метод тяг Н.Е. Жуковского для горизонтального полёта:

$$P \cdot \cos(90° + \alpha - \varepsilon) - c_x(\alpha)\frac{\rho V^2}{2}S = 0;$$

$$P \cdot \sin(90° + \alpha - \varepsilon) + c_y(\alpha)\frac{\rho V^2}{2}S = m \cdot g;$$

Исходя из системы:

$$\varepsilon = -\operatorname{atan}\left(\frac{m \cdot g - c_y(\alpha)\frac{\rho V^2}{2}S}{c_x(\alpha)\frac{\rho V^2}{2}S}\right) \cdot \left(\frac{180°}{\pi}\right) - 90° - \alpha;$$

$$\sum_{i=1}^{4} P = \frac{c_x(\alpha)\frac{\rho V^2}{2}S}{\cos(90° + \alpha - \varepsilon)}$$

Ответ: $\boldsymbol{\varepsilon_1 = \varepsilon_2 = 27.81°}$, $\boldsymbol{\sum_{i=1}^{4} P = 435{,}95}$ Н

Задача № 4

Математическое моделирование динамики полёта БВС типа конвертоплан позволяет решить задачи разработки алгоритмов его управления на ранних этапах проектирования. В рамках данной задачи, будет рассмотрен продольный канал управления объектом типа конвертоплан выполняющего установившийся горизонтальный полёт: необходимо написать алгоритм управления углом тангажа БВС типа конвертоплан при помощи аэродинамических поверхностей (элеронов заднего крыла).

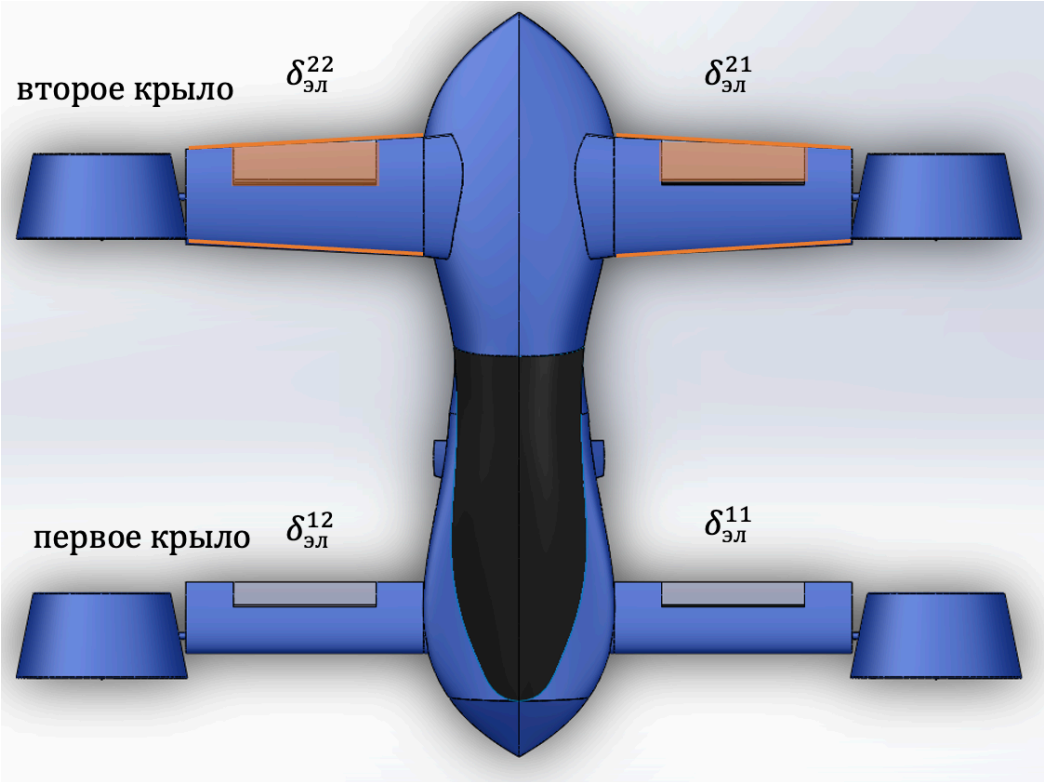


Рисунок 1. Управляющие поверхности БВС.

Участникам выдаётся проект математической модели БВС, написанный на языке Python, который состоит из следующих файлов:

- *UAV.py* – содержит в себе класс – реализацию математической модели БВС типа конвертоплан, осуществляющий движение в продольном канале. **Данный файл изменять не нужно.**
- *SAU_template.py* – класс управления БВС. **Решение задачи реализовывать в данном файле.**
- *Simulate_n_result.ipynb* – файл Jupyter Notebook, при помощи которого осуществляется запуск проекта. **Выполнение моделирования производится при помощи выполнения ячейки «Нелинейная модель»**, предварительно необходимо вызвать ячейку «Исходные данные», построить графики моделирования возможно при помощи вызова ячейки «Визуализация результатов».
- *AeroStruct.mat* – файл содержит в себе коэффициенты полиномов аэродинамических зависимостей. Данный файл должен лежать в корне проекта.
- *results_uav.xlsx* – файл содержит в себе запись результатов моделирования, считывается при вызове программы Tool.py.
- *Tool.py* – содержит в себе функцию оценки переходного процесса, данную программу можно использовать для проверки решения.

В данной постановке задачи управление в продольном канале можно осуществлять при помощи создания аэродинамического момента M_z^a путём отклонения элеронов второго крыла $\delta_{эл}^{21}, \delta_{эл}^{22}$, используя его как руль высоты, при этом отклонения элеронов первого крыла $\delta_{эл1} = \delta_{эл}^{11} = \delta_{эл}^{12} = 0°$, а элероны второго крыла отклоняются синхронно: $\delta_{эл2} = \delta_{эл}^{21} = \delta_{эл}^{22} = \delta_{эл}^{зад}$. Углы отклонения элеронов лежат в диапазоне $-25° \leq \delta_{эл2} \leq 25°$, как показано на рисунке 2.

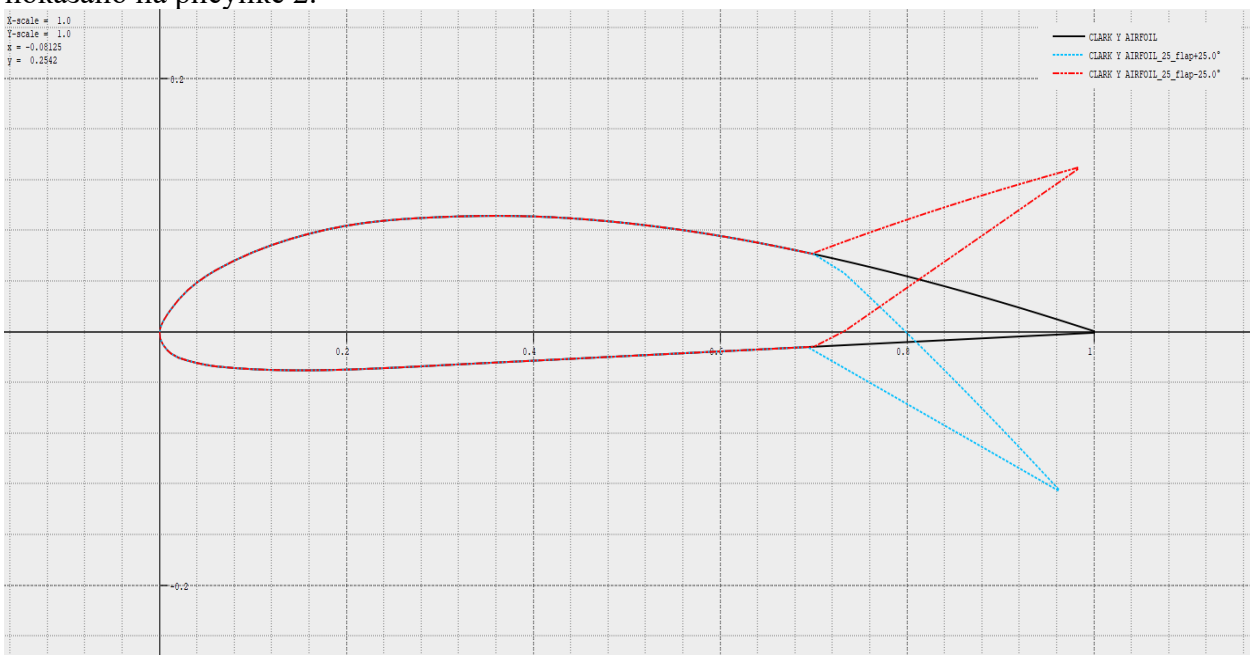


Рисунок 2. Профиль второго крыла и углы отклонения элеронов в положении $-25°$, $0°$, $-25°$.

В связи с этим уравнение аэродинамических моментов, создаваемых элеронами второго крыла, можно выразить как:

$$M_z^a = m_z(\alpha, \delta_{эл2}) \cdot \frac{\rho \cdot V^2}{2} \cdot S \cdot l.$$

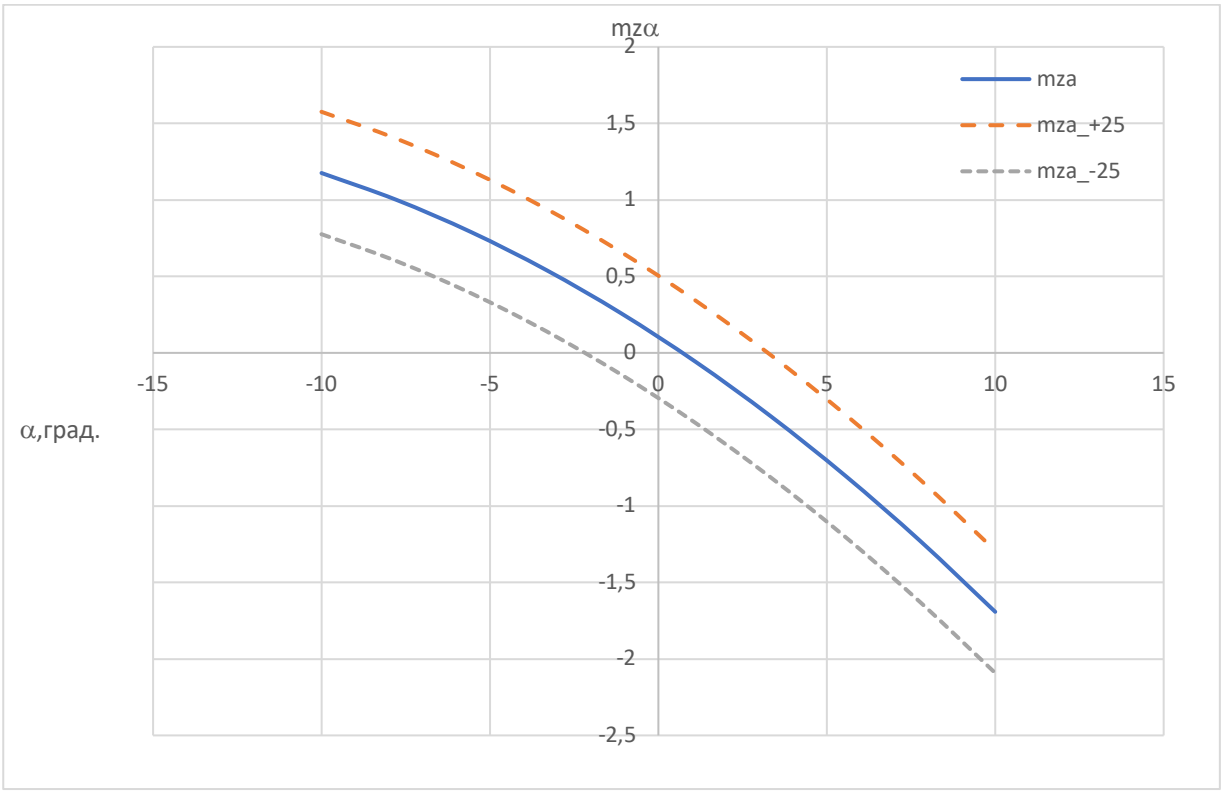


Рисунок 3. Изменение аэродинамического момента $m_z(\alpha, \delta_{эл2})$ при отклонении элеронов второго крыла.

Разрабатываемый алгоритм системы автоматического управления (САУ) БВС вычисляется каждый такт работы вычислителя автопилота. САУ получает истинные значения от системы бортовых измерений (СБИ) БВС: угол тангажа ϑ – град., угловую скорость по оси O_{Z1} ω_{z1} – град./с, вертикальную скорость V_y – м/с, геометрическую высоту H – м. На выходе алгоритма необходимо определить заданные значения углов отклонения элеронов второго крыла $\delta_{эл2}$ – град.

Реализовать алгоритм возможно при помощи пропорционально-интегрально-дифференциального регулятора (ПИД-регулятора), структурная схема ПИД-регулятора показана на рисунке 4.

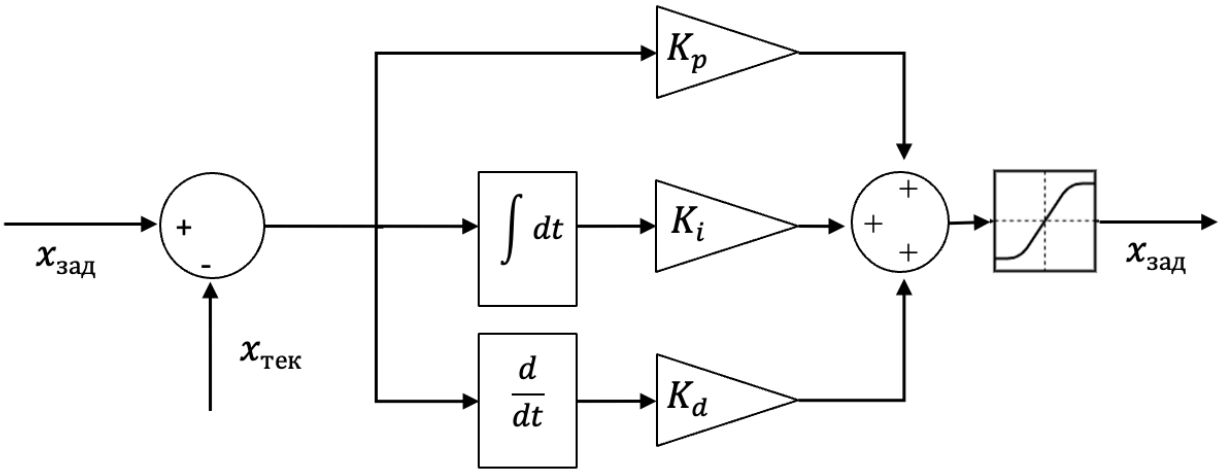


Рисунок 4. Структурная схема ПИД Регулятора

Как показано на рисунке 4, на вход регулятора подаётся значение ошибки регулируемой величины (разница заданного и текущего значений). Эта ошибка поступает

на вход одного, двух, или трёх звеньев: пропорционального, интегрального и дифференциального (в зависимости от сложности регулирования). Пропорциональное звено умножает (усиливает) значение ошибки на некоторый постоянный коэффициент K_p, а дифференциальное звено умножает производную ошибки ($\frac{d}{dt}$) на коэффициент K_d, интегральное звено умножает нарастающую ошибку на коэффициент K_i для «подтягивания» регулируемого значения ближе к заданному (устранение статической ошибки).

Выходной сигнал регулятора, как правило, ограничивают в допустимых пределах (блок насыщения). Представленную структуру можно записать в виде формулы:

$$x_{\text{упр}} = (x_{\text{зад}} - x_{\text{тек}}) \cdot k_p + \int (x_{\text{зад}} - x_{\text{тек}})dt \cdot k_i + \frac{d(x_{\text{зад}} - x_{\text{тек}})}{dt} \cdot k_d.$$

В общем виде, переходный процесс регулируемой величины показан на рисунке 5.

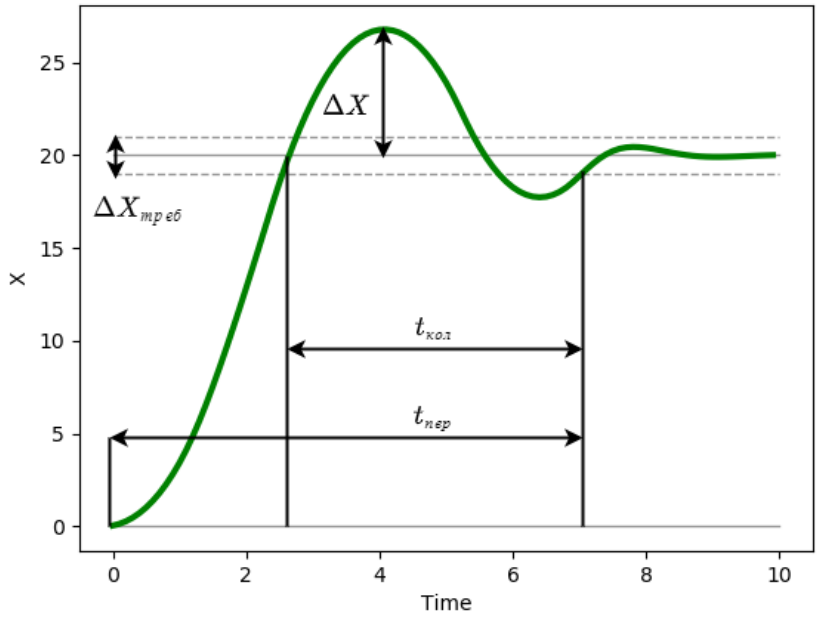


Рисунок 5. Переходный процесс регулируемой величины.

На рисунке обозначены:

- трубка точности $\Delta X_{\text{треб}}$ – допустимое граничное значение ошибки регулируемой величины;
- время переходного процесса $t_{\text{пер}}$ – процесс считается завершённым, когда регулируемая величина попадает в заданную трубку точности и больше не выходит за её пределы;
- время затухания $t_{\text{кол}}$ – время затухания колебаний после достижения требуемого значения (как правило, этот параметр зависит от величины дифференциального коэффициента k_d);
- перерегулирование ΔX – отклонение от заданного значения величины в противоположную сторону (как правило, этот параметр зависит от величины пропорционального коэффициента k_p).

Напишите программу на языке программирования Python, реализующую ПИД или любой из его вариаций регулятор высоты $H^{\text{зад}}$ и регулятор приборной скорости $V_{\text{пр}}^{\text{зад}}$. Характеристики переходных процессов заданных величин должны находиться в рамках требований к переходным процессам.

БВС осуществляет полёт в самолётном режиме на высоте H_0(м), с приборной скоростью $V_{\text{пр}0}$(км/ч), углы поворота ВМГ $\varepsilon_1, \varepsilon_2 = 90°$ зафиксированы, значения отклонений элеронов первого крыла $\delta_{\text{эл}1} = \delta_{\text{эл}}^{11} = \delta_{\text{эл}}^{12} = 0°$.

Шаблоном программы является класс SAU, включённый в моделирования посредством классов-интерфейсов SAU_in и SAU_out.

Код и структура класса SAU_in (на языке Python) имеют вид:

```
class SAU_in:
    # Текущий угол наклона траектории, град.
    Tet = 0
    # Текущий угол тангажа, град.
    Tan = 0
    # Текущая угловая скорость, связанная СК, град/сек
    Wz1 = 0
    # Текущая высота, м.
    H = 0
    # Текущая вертикальная скорость,м/с
    Vy = 0
    # Текущая приборная скорость, км/ч
    Vw = 0
```

Код и структура класса SAU_out (на языке Python) имеют вид:

```
class SAU_out:
    # Заданные значения положения дросселя для ВМГ 1,2,  от 0.1
до 1
    P12_dr = 0.2
    # Заданные значения положения дросселя для ВМГ 3,4, от 0.1
до 1
    P34_dr = 0.2
    # Заданное значение угла поворота ВМГ первого и второго
крыла, град
    eps = 90.0
    # Заданное значения угла поворота элеронов второго крыла
    de = 0.0
```

Код и структура класса SAU (на языке Python) имеют вид:

```
# Функция ограничения значения
clamp = lambda n, minn, maxn: max(min(maxn, n), minn)

class SAU:
    def __init__(self,H_zad,Vpr_zad,eps,L_zad):
        self.u_output = SAU_out()
        self.u_output.eps = eps
        # Шаг интегрирования
        self.dt = 0.01
        # Время моделирования
        self.Time = 0
        # Заданный угол тангажа
        self.TangU = 0
        # Производная заданного угла тангажа
        self.TangUDt = 0
        # Интеграл ошибки
```

```
        self.Integral_H = 0
        # Заданное значение положения дросселя для ВМГ 1,2
        self.P_reg_12 = 0
        # Заданное значение положения дросселя для ВМГ 3,4
        self.P_reg_34 = 0
        # Заданный угол отклонения элеронов второго крыла
sigma_3,sigma_4
        self.delta_elerons = 0

        # Заданные регулируемые значения
        # Заданная высота, м.
        self.H_zad = H_zad
        # Заданная приборная скорость, км/ч
        self.Vpr_zad = Vpr_zad
        # Заданное значение
        self.L_zad = L_zad

        # Плечи
        self.lz1 = 1
        self.lz2 = 0.84
        self.ly1 = 0.036
        self.ly2 = 0.14

        # Масса БВС
        self.m = 108

        # Для получения дроссельных характеристик в зависимости
от расчетной тяги

        self.drossel = np.linspace(0, 1, 10, endpoint=True) #
Положение дросселя, 0..1
        self.P_one = np.linspace(0, 350, 10, endpoint=True) #
Тяга,Ньютон

        # Шаблон записи для отладки
        self.writenames
=list(["Time","TanU","TanU_dt","P_reg_12","P_reg_34"])
        data_sau =
[self.Time,self.TangU,self.TangUDt,self.P_reg_12,self.P_reg_34]

        self.db = DataFrame([data_sau],columns =
self.writenames)
    # Функция записи данных
    def writeframe(self):
        data_sau =
[self.Time,self.TangU,self.TangUDt,self.P_reg_12,self.P_reg_34]
        frame = DataFrame([data_sau],columns = self.writenames)
        self.db = concat([self.db, frame], ignore_index=True)

    # Шаблонная функция вычисления заданного положения дросселя
для регулирования приборной скорости
    def calc_drossel(self):
```

```
        # Заданные значения
        # Заданная приборная скорость, км/ч
        Vpr_zad = self.Vpr_zad/3.6
        # Параметры которые могут понадобится для расчета
        # Текущая приборная скорость, км/ч
        Vpr_now =  self.u_input.Vw/3.6


        self.P_reg = 0.1
        # Ограничение [0.1,1] обязательно
        self.drossel = clamp(self.P_reg,0.1,1)
        self.P_reg_12  =  self.drossel*0.5
        self.P_reg_34  =  self.drossel*0.5

    # Шаблонная функция вычисления заданного отклонения
элеронов второго крыла для регулирования высоты
    def calc_delta_elerons(self):
        # Заданные значения
        H_zad = self.H_zad
        # Параметры которые могут понадобится для расчета
        # Текущий угол тангажа, град.
        Tang = self.u_input.Tan
        # Текущая угловая скорость по оси OZ, в связанной СК
        Wz1 = self.u_input.Wz1
        # Текущая высота БВС, м.
        H_now = self.u_input.H
        # Текущая вертикальная скорость БВС, м.
        Vy = self.u_input.Vy

        delta_elerons_calc =0

        # Ограничение -25 +25 обязательно
        self.delta_elerons = clamp(delta_elerons_calc,-25,25)

    # Основная функция расчета, вызывается при моделировании
    def calc_PID(self,input:SAU_in):

        self.u_input = input
        # Вызов шаблонной функции, расчета дросселя
        self.calc_drossel()
        # Вызов шаблонной функции, расчета элеронов заднего
крыла
        self.calc_delta_elerons()
        # Запись параметров на мат. модель
        self.u_output.de = self.delta_elerons  # Заданное
значение положения элеронов второго крыла
        self.u_output.P12_dr = self.P_reg_12    # Заданное
значение дросселя ВМГ 1-2
        self.u_output.P34_dr = self.P_reg_34    # Заданное
значение дросселя ВМГ 1-2
        self.Time += self.dt
        self.writeframe()
```

```
        return self.u_output

    def drop_u(self):
        return self.u_output
```

Рекомендуется производить вычисления в функциях-шаблонах:

- calc_delta_elerons – шаблонная функция вычисления заданного отклонения элеронов второго крыла для регулирования высоты;
- calc_drossel – шаблонная функция вычисления заданного положения дросселя для регулирования приборной скорости.

Условия моделирования полёта БВС:

- начальное значение угла тангажа Tang = 1,5 град.;
- начальное значение угла наклона траектории Theta = 0 град.;
- начальное значение угла атаки Alpha = 1,5 град.;
- начальное значение земной скорости Va = 55,5 м/с;
- начальное значение приборной скорости Vpr = 200 км/ч;
- начальное значение высоты БВС H = 1000 м.

Заданные параметры полета БВС:

- заданное значение высоты БВС H = 1100 м;
- заданное значение приборной скорости БВС Vpr = 200 км/ч.

Требования к переходному процессу по высоте H и приборной скорости Vpr:

	Высокая точность	Средняя точность	Низкая точность
Статическая ошибка	$\Delta X_{треб} \le 2{,}3$	$2{,}3 < \Delta X_{треб} \le 2{,}7$	$\Delta X_{треб} > 2{,}7$
Время переходного процесса	$t_{пер} \le 11{,}5$	$11{,}5 < t_{пер} \le 13{,}5$	$t_{пер} > 13{,}5$
Перерегулирование	$\Delta X \le 11{,}5$	$11{,}5 < \Delta X \le 13{,}5$	$\Delta X > 13{,}5$

Решение:

```
from UAV import SAU_in,SAU_out
from math import *
from pandas import DataFrame, concat
import numpy as np
# Функция ограничения значения
clamp = lambda n, minn, maxn: max(min(maxn, n), minn)

class SAU:
    def __init__(self,H_zad,Vpr_zad,eps,L_zad):
        self.u_output = SAU_out()
        self.u_output.eps = eps
        # Шаг интегрирования
```

```
        self.dt = 0.01
        # Время моделирования
        self.Time = 0
        # Заданный угол тангажа
        self.TangU = 0
        # Производная заданного угла тангажа
        self.TangUDt = 0
        # Интеграл ошибки
        self.Integral_H = 0
        # Заданное значение положения дросселя для ВМГ 1,2
        self.P_reg_12 = 0
        # Заданное значение положения дросселя для ВМГ 3,4
        self.P_reg_34 = 0
        # Заданный угол отклонения элеронов второго крыла
sigma_3,sigma_4
        self.delta_elerons = 0

        # Заданные регулируемые значения
        # Заданная высота, м.
        self.H_zad = H_zad
        # Заданная приборная скорость, км/ч
        self.Vpr_zad = Vpr_zad
        # Заданное значение
        self.L_zad = L_zad

        # Плечи
        self.lz1 = 1
        self.lz2 = 0.84
        self.ly1 = 0.036
        self.ly2 = 0.14

        # Масса БВС
        self.m = 108

        # Для получения дроссельных характеристик в зависимости
от расчетной тяги

        self.drossel = np.linspace(0, 1, 10, endpoint=True) #
Положение дросселя, 0..1
        self.P_one = np.linspace(0, 350, 10, endpoint=True) #
Тяга,Ньютон

        # Шаблон записи для отладки
        self.writenames
=list(["Time","TanU","TanU_dt","P_reg_12","P_reg_34"])
        data_sau =
[self.Time,self.TangU,self.TangUDt,self.P_reg_12,self.P_reg_34]

        self.db = DataFrame([data_sau],columns =
self.writenames)
    # Функция записи данных
    def writeframe(self):
```

```
        data_sau =
[self.Time,self.TangU,self.TangUDt,self.P_reg_12,self.P_reg_34]
        frame = DataFrame([data_sau],columns = self.writenames)
        self.db = concat([self.db, frame], ignore_index=True)



    # Шаблонная функция вычисления заданного положения дросселя
для регулирования приборной скорости
    def calc_drossel(self):
        # Заданные значения
        # Заданная приборная скорость, км/ч
        Vpr_zad = self.Vpr_zad/3.6
        # Параметры которые могут понадобится для расчета
        # Текущая приборная скорость, км/ч
        Vpr_now =  self.u_input.Vw/3.6
        # Расчет
        k_P = 0.5 # Пропорциональный коэффициент
        self.P_reg = 0.1 + k_P*(Vpr_zad - Vpr_now)
        # Ограничение [0.1,1] обязательно
        self.drossel = clamp(self.P_reg,0.1,1)
        self.P_reg_12  =  self.drossel*0.5
        self.P_reg_34  =  self.drossel*0.5

    # Шаблонная функция вычисления заданного отклонения
элеронов второго крыла для регулирования высоты
    def calc_delta_elerons(self):
        # Заданные значения
        H_zad = self.H_zad
        # Параметры которые могут понадобится для расчета
        # Текущий угол тангажа, град.
        Tang = self.u_input.Tan
        # Текущая угловая скорость по оси OZ, в связанной СК
        Wz1 = self.u_input.Wz1
        # Текущая высота БВС, м.
        H_now = self.u_input.H
        # Текущая вертикальная скорость БВС, м.
        Vy = self.u_input.Vy
        # Эталонный расчет

        # Пропорциональное звено
        k_P = 5.5
        # Интегральное звено
        k_i = 0.1
        # Дифференциальное звено
        k_d = 1.8

        # Заданный угол тангажа в зависимости от высоты, с
ограничением
        self.TangU = clamp(-1.0 * (H_now - H_zad) - 0.5 * Vy,-
20,20)
```

```
        delta_elerons_calc = k_P*( self.TangU  - Tang ) - k_d *
Wz1 - k_i * self.Integral_H
        # Интеграл ошибки
        self.Integral_H += (H_now-H_zad)*self.dt
        # Ограничение -25 +25 обязательно
        self.delta_elerons = clamp(delta_elerons_calc,-25,25)

    # Основная функция расчета, вызывается при моделировании
    def calc_PID(self,input:SAU_in):

        self.u_input = input
        # Вызов шаблонной функции, расчета дросселя
        self.calc_drossel()
        # Вызов шаблонной функции, расчета элеронов заднего
крыла
        self.calc_delta_elerons()
        # Запись параметров на мат. модель
        self.u_output.de = self.delta_elerons  # Заданное
значение положения элеронов второго крыла
        self.u_output.P12_dr = self.P_reg_12    # Заданное
значение дросселя ВМГ 1-2
        self.u_output.P34_dr = self.P_reg_34    # Заданное
значение дросселя ВМГ 1-2
        self.Time += self.dt
        self.writeframe()
        return self.u_output

    def drop_u(self):
        return self.u_output
```

Вариант № 2

Задача № 1

Рассчитайте размах крыла, длины корневой и концевой хорд трапециевидного крыла БЛА самолётного типа, если площадь крыла 8 м^2, сужение 3, удлинение 9. Округлите ответ до десятых.

Решение (основные вычисления):

$$\text{Размах} = \sqrt{\text{Удлинение} * \text{Площадь}} = \sqrt{9 * 8\text{м}^2} = 8{,}5 \text{ м}$$

$$\text{Корневая хорда} = \frac{2 * \text{сужение} * \text{площадь}}{\text{размах} * (1 + \text{сужение})} = \frac{2 * 3 * 8 \text{ м}^2}{8{,}5 * (1 + 3)} = 1{,}4$$

$$\text{Концевая хорда} = \frac{\text{корневая хорда}}{\text{сужение}} = \frac{1{,}4}{3} = 0{,}5$$

Ответ: размах крыла 8,5 м; корневая хорда 1,4 м; концевая хорда 0,5 м.

Задача № 2

Для работы наземных систем обслуживания БЛА применяют различные варианты электродвигателей. Асинхронный электродвигатель мощностью 1 кВт подключен к трёхфазной сети с линейным напряжением сети 380 В, и частотой 50 Гц. Определите амплитуду тока в фазных проводах, если коэффициент мощности двигателя (cos φ) равен 0,7. Ответ дать с точностью до сотых.

Решение (основные вычисления):

Полная мощность определяется с учётом симметричности линейной нагрузки: $P = \sqrt{3} \cdot U \cdot I \cdot cos\varphi$. Тогда действующее значение тока в каждой фазе составит:

$$I = \frac{P}{\sqrt{3} \cdot U \cdot I \cdot cos\varphi} = \frac{1000}{\sqrt{3} \cdot 380 \cdot 0{,}7} = 2{,}17 \text{ А}.$$

Амплитудное значение силы в фазных проводах:

$$I = \frac{I_m}{\sqrt{2}} \text{ тогда } I_m = \sqrt{2} \cdot I = \sqrt{2} \cdot 2{,}17 = 3{,}07 \text{ А}.$$

Ответ: амплитудная величина тока в каждой фазе I_m= 3,07 А.

Задача № 3

Управляемый вектор тяги беспилотного воздушного судна (БВС) схемы тандем (далее – конвертоплан) (рисунок 1) позволяет совершать прямолинейный горизонтальный полёт, сбалансированный при помощи аэродинамических сил и направления тяги винтомоторной группы (ВМГ).

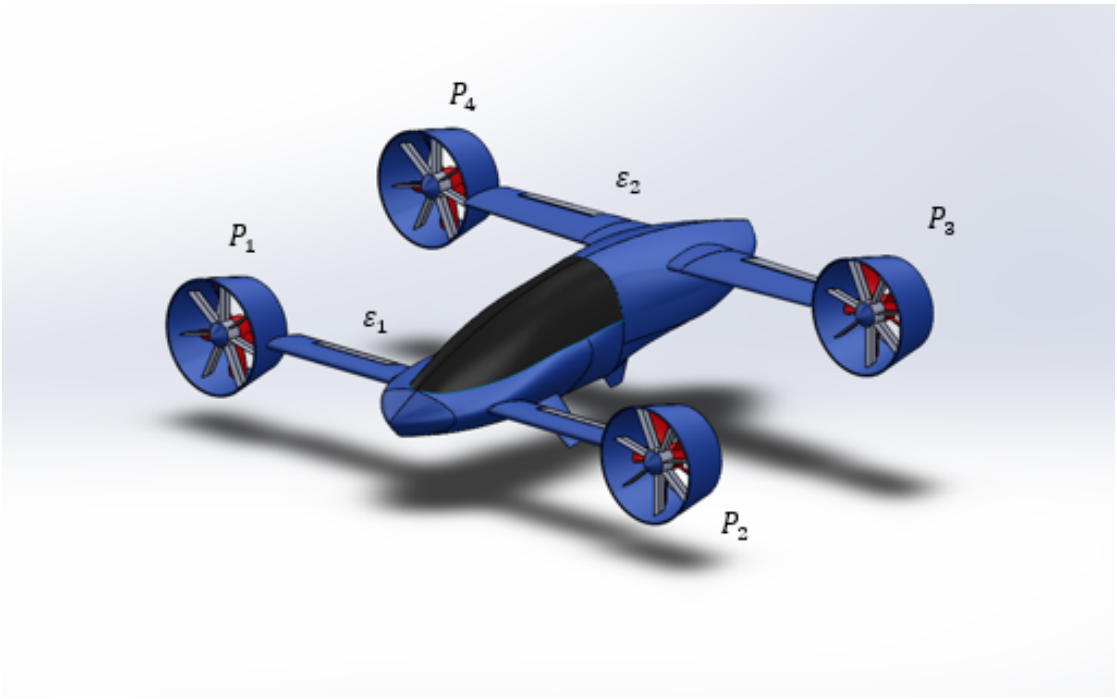
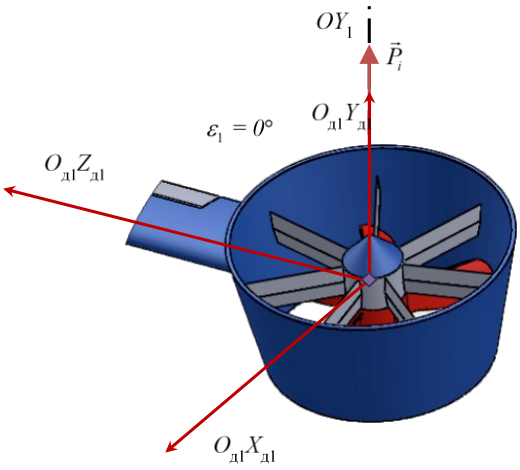


Рисунок 1. Внешний облик БВС, конвертоплан схемы тандем.

Угловые положения осей вращения двигателей (сил тяги $\vec{P}_i$) определяются углами поворота этих осей вокруг поворотных осей двигателей $O_{д1}Z_{д1}$, $O_{д2}Z_{д2}$ на угол ε_1 для первой пары и вокруг осей $O_{д3}Z_{д3}$, $O_{д4}Z_{д4}$ на угол ε_2 для второй пары соответственно.



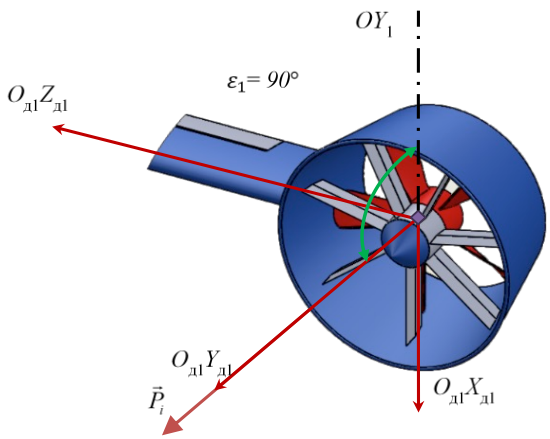


Рисунок 2. Вращение ВМГ вокруг оси ε_j.

В установившимся режиме полёта сумма моментов, действующих на БВС, равна нулю, и он находится в состоянии балансировки – для дальнейших расчётов необходимо воспользоваться уравнениями его установившегося движения. Схема сил, действующих на БВС в самолётном режиме, приведена на рисунке 3.

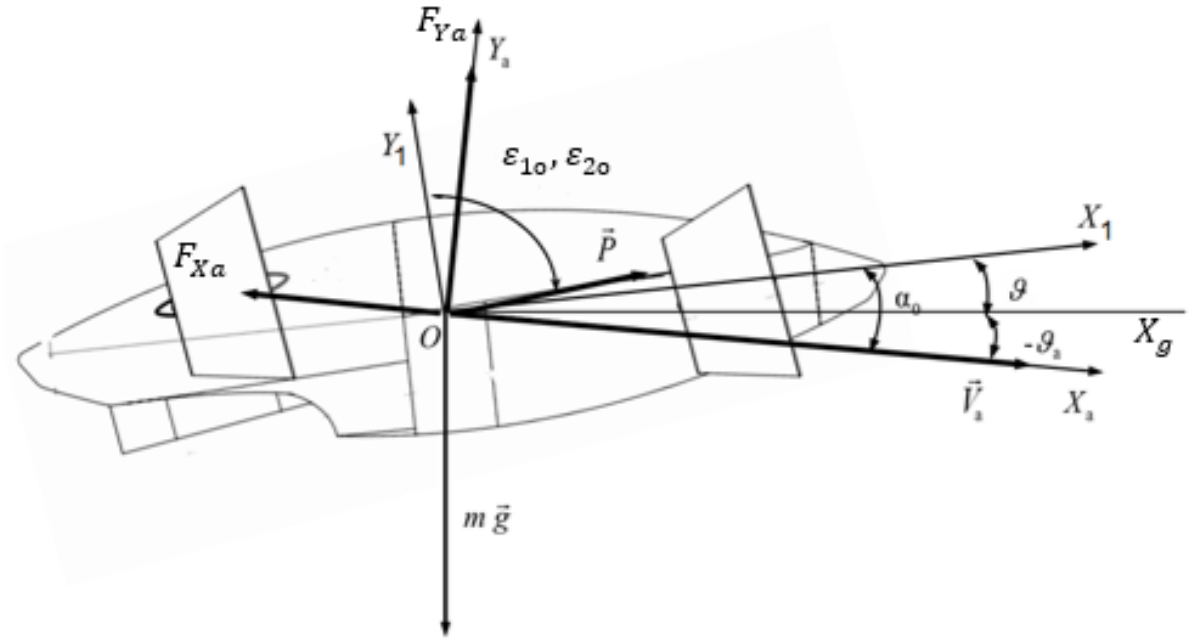


Рисунок 3. Силы, действующие на БВС в самолётном режиме.

Сила лобового сопротивления F_{Xa} направлена противоположно вектору воздушной скорости V_a БВС и равна:

$$F_{Xa} = c_x(\alpha)\frac{\rho V^2}{2}S.$$

Подъёмная сила F_{Ya} направлена перпендикулярно вектору воздушной скорости V_a БВС и равна:

$$F_{Ya} = c_y(\alpha)\frac{\rho V^2}{2}S.$$

Вектор силы тяжести направлен от центра масс БВС перпендикулярно вниз:

$$G = m \cdot g.$$

Конвертоплан совершает установившийся горизонтальный прямолинейный полёт (V = const, $\theta = 0$, H = const). Определите величину углов отклонения оси винтомоторной группы $\varepsilon_1, \varepsilon_2$, а также сумму тяг ВМГ $\sum_{i=1}^{4} P$, при условии, что $\varepsilon_1 = \varepsilon_2$. При решении данной задачи вращением БВС вокруг центра масс пренебречь. Ответ указать в градусах и Н с точностью до сотых.

При расчётах принять следующие параметры:

- ускорение свободного падения $g = 9{,}806$ м/с^2;
- плотность воздушной среды на текущей высоте $\rho = 1{,}2135$ кг/м^3;
- масса БВС $m = 108$ кг;
- характерная площадь крыла $S = 2{,}18$ м^2;
- истинная скорость $V_a = 50$ м/с;

- угол наклона траектории $\theta = 0°$;
- угол атаки $\alpha = 3{,}6°$;
- коэффициент лобового сопротивления $c_x(\alpha) = 0{,}0769$;
- коэффициент подъемной силы $c_y(\alpha) = 0{,}277$.

Решение (основные вычисления):

Метод тяг Н.Е. Жуковского для горизонтального полёта:

$$P \cdot \cos(90° + \alpha - \varepsilon) - c_x(\alpha)\frac{\rho V^2}{2}S = 0;$$

$$P \cdot \sin(90° + \alpha - \varepsilon) + c_y(\alpha)\frac{\rho V^2}{2}S = m \cdot g;$$

Исходя из системы:

$$\varepsilon = -\operatorname{atan}\left(\frac{m \cdot g - c_y(\alpha)\frac{\rho V^2}{2}S}{c_x(\alpha)\frac{\rho V^2}{2}S}\right) \cdot \left(\frac{180°}{\pi}\right) - 90° - \alpha;$$

$$\sum_{i=1}^{4} P = \frac{c_x(\alpha)\frac{\rho V^2}{2}S}{\cos(90° + \alpha - \varepsilon)}$$

Ответ: $\boldsymbol{\varepsilon_1 = \varepsilon_2 = 64.52°}$, $\boldsymbol{\sum_{i=1}^{4} P = 291.18}$ Н.

Задача № 4

Математическое моделирование динамики полёта БВС типа конвертоплан позволяет решить задачи разработки алгоритмов его управления на ранних этапах проектирования. В рамках данной задачи, будет рассмотрен продольный канал управления объектом типа конвертоплан выполняющего установившийся горизонтальный полёт: необходимо написать алгоритм управления углом тангажа БВС типа конвертоплан при помощи аэродинамических поверхностей (элеронов заднего крыла).

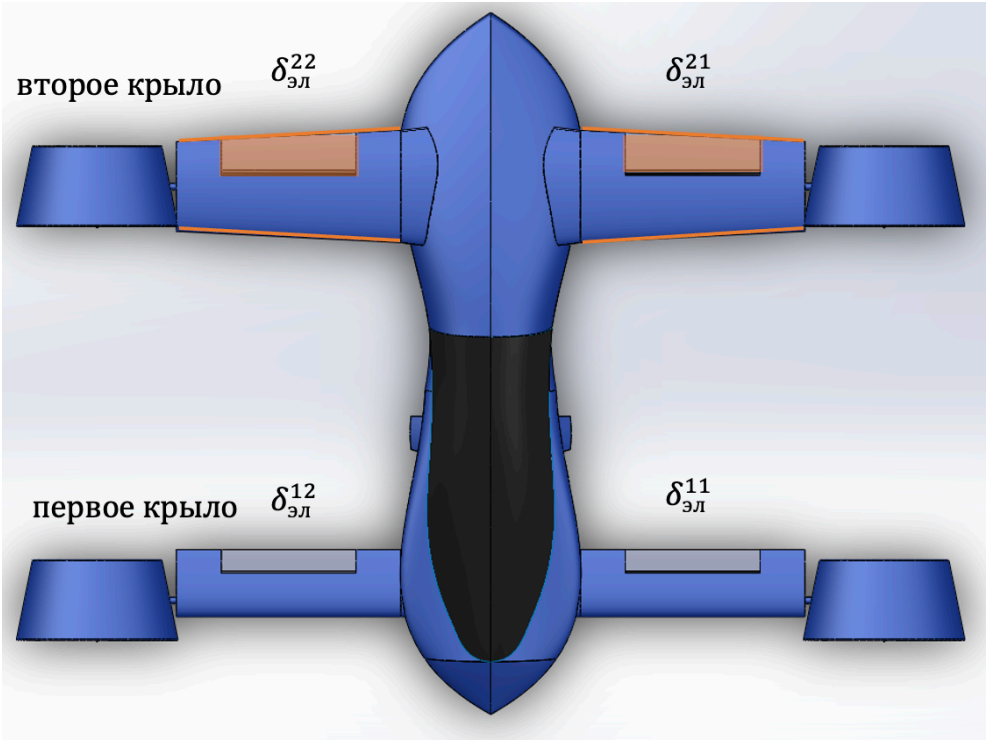


Рисунок 1. Управляющие поверхности БВС.

Участникам выдаётся проект математической модели БВС, написанный на языке Python, который состоит из следующих файлов:

- *UAV.py* – содержит в себе класс – реализацию математической модели БВС типа конвертоплан, осуществляющий движение в продольном канале. **Данный файл изменять не нужно.**
- *SAU_template.py* – класс управления БВС. **Решение задачи реализовывать в данном файле.**
- *Simulate_n_result.ipynb* – файл Jupyter Notebook, при помощи которого осуществляется запуск проекта. **Выполнение моделирования производится при помощи выполнения ячейки «Нелинейная модель»**, предварительно необходимо вызвать ячейку «Исходные данные», построить графики моделирования возможно при помощи вызова ячейки «Визуализация результатов».
- *AeroStruct.mat* – файл содержит в себе коэффициенты полиномов аэродинамических зависимостей. Данный файл должен лежать в корне проекта.
- *results_uav.xlsx* – файл содержит в себе запись результатов моделирования, считывается при вызове программы Tool.py.
- *Tool.py* – содержит в себе функцию оценки переходного процесса, данную программу можно использовать для проверки решения.

В данной постановке задачи управление в продольном канале можно осуществлять при помощи создания аэродинамического момента M_z^a путём отклонения элеронов второго крыла $\delta_{эл}^{21}, \delta_{эл}^{22}$, используя его как руль высоты, при этом отклонения элеронов первого крыла $\delta_{эл1} = \delta_{эл}^{11} = \delta_{эл}^{12} = 0°$, а элероны второго крыла отклоняются синхронно: $\delta_{эл2} = \delta_{эл}^{21} = \delta_{эл}^{22} = \delta_{эл}^{зад}$. Углы отклонения элеронов лежат в диапазоне $-25° \leq \delta_{эл2} \leq 25°$, как показано на рисунке 2.

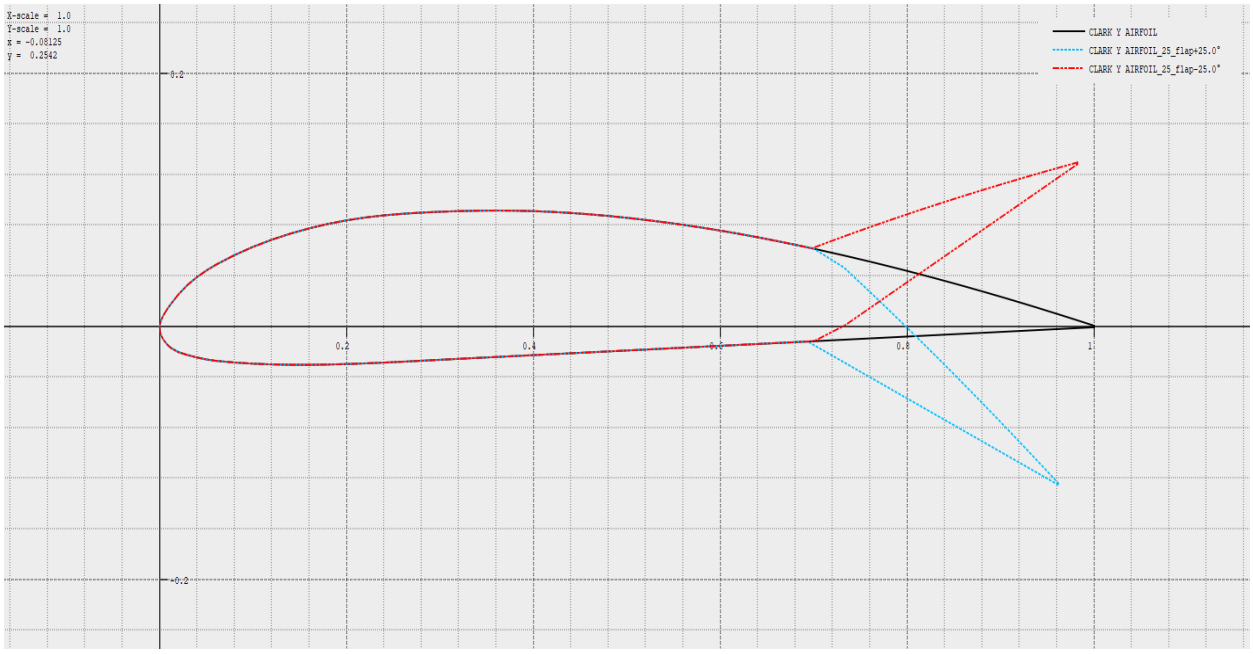


Рисунок 2. Профиль второго крыла и углы отклонения элеронов в положении $-25°$, $0°$, $-25°$.

В связи с этим уравнение аэродинамических моментов, создаваемых элеронами второго крыла, можно выразить как:

$$M_z^a = m_z(\alpha, \delta_{эл2}) \cdot \frac{\rho \cdot V^2}{2} \cdot S \cdot l.$$

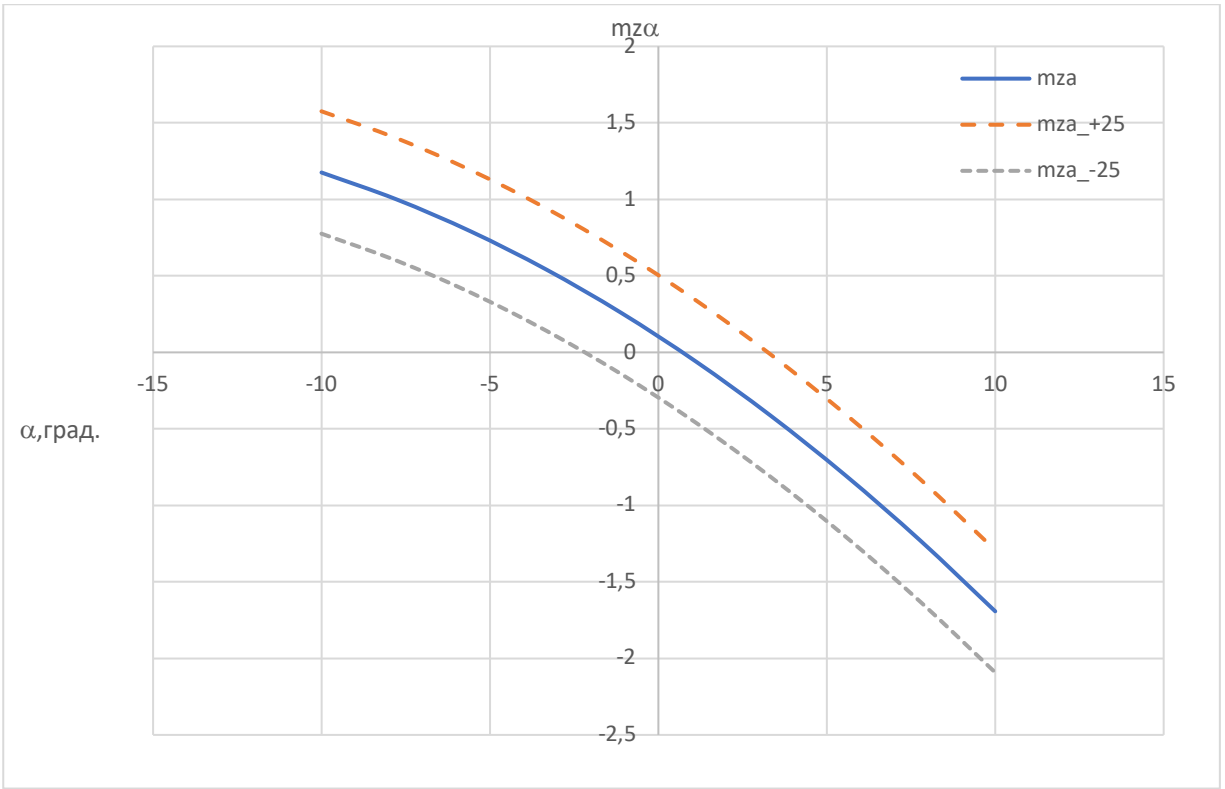


Рисунок 3. Изменение аэродинамического момента $m_z(\alpha, \delta_{эл2})$ при отклонении элеронов второго крыла.

Разрабатываемый алгоритм системы автоматического управления (САУ) БВС вычисляется каждый такт работы вычислителя автопилота. САУ получает истинные значения от системы бортовых измерений (СБИ) БВС: угол тангажа ϑ – град., угловую скорость по оси O_{Z1} ω_{z1} – град./с, вертикальную скорость V_y – м/с, геометрическую высоту H – м. На выходе алгоритма необходимо определить заданные значения углов отклонения элеронов второго крыла $\delta_{эл2}$ – град.

Реализовать алгоритм возможно при помощи пропорционально-интегрально-дифференциального регулятора (ПИД-регулятора), структурная схема ПИД-регулятора показана на рисунке 4.

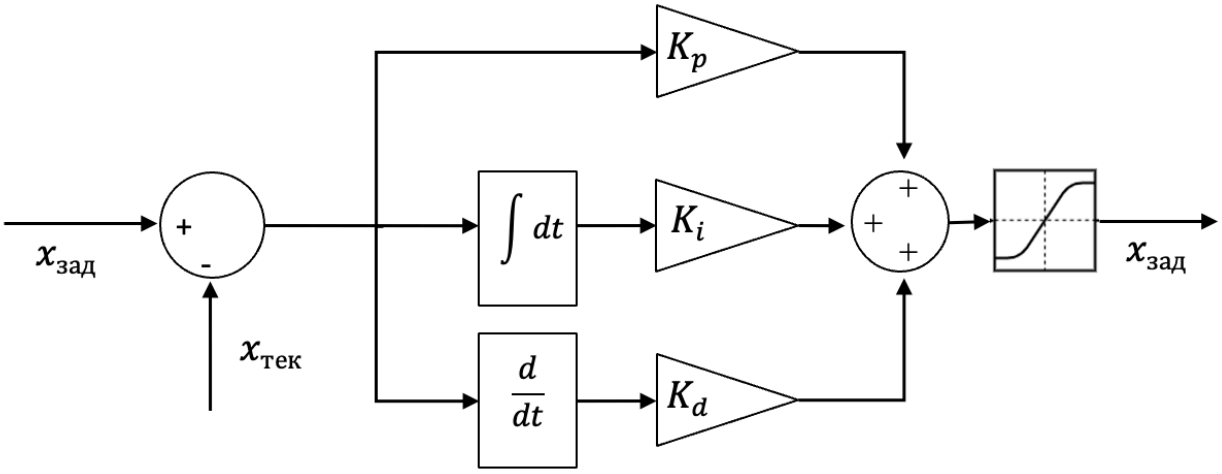


Рисунок 4. Структурная схема ПИД Регулятора

Как показано на рисунке 4, на вход регулятора подаётся значение ошибки регулируемой величины (разница заданного и текущего значений). Эта ошибка поступает на вход одного, двух, или трёх звеньев: пропорционального, интегрального и дифференциального (в зависимости от сложности регулирования). Пропорциональное звено умножает (усиливает) значение ошибки на некоторый постоянный коэффициент K_p, а дифференциальное звено умножает производную ошибки ($\frac{d}{dt}$) на коэффициент K_d, интегральное звено умножает нарастающую ошибку на коэффициент K_i для «подтягивания» регулируемого значения ближе к заданному (устранение статической ошибки).

Выходной сигнал регулятора, как правило, ограничивают в допустимых пределах (блок насыщения). Представленную структуру можно записать в виде формулы:

$$x_{упр} = (x_{зад} - x_{тек}) \cdot k_p + \int (x_{зад} - x_{тек}) dt \cdot k_i + \frac{d(x_{зад} - x_{тек})}{dt} \cdot k_d.$$

В общем виде, переходный процесс регулируемой величины показан на рисунке 5.

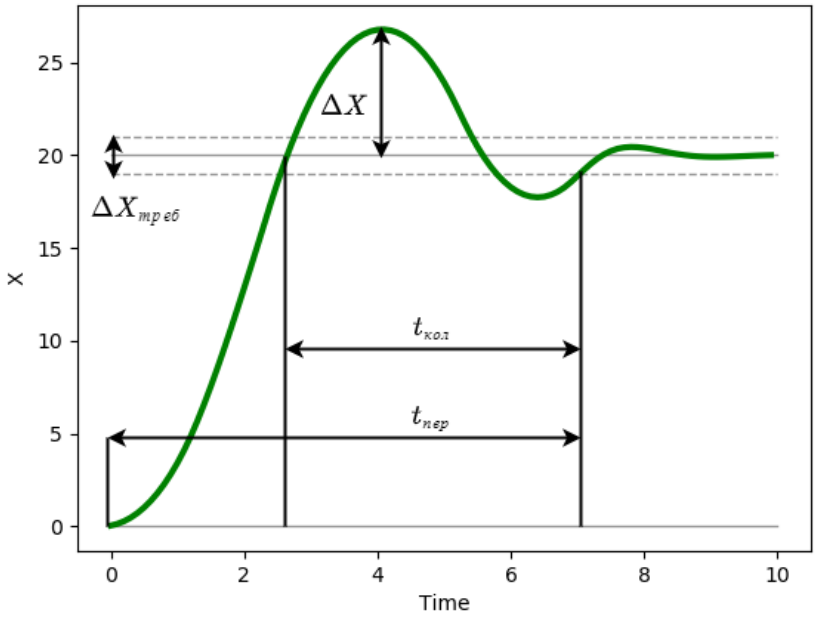


Рисунок 5. Переходный процесс регулируемой величины.

На рисунке обозначены:

- трубка точности $\Delta X_{\text{треб}}$ – допустимое граничное значение ошибки регулируемой величины;
- время переходного процесса $t_{\text{пер}}$ – процесс считается завершённым, когда регулируемая величина попадает в заданную трубку точности и больше не выходит за её пределы;
- время затухания $t_{\text{кол}}$ – время затухания колебаний после достижения требуемого значения (как правило, этот параметр зависит от величины дифференциального коэффициента k_d);
- перерегулирование ΔX – отклонение от заданного значения величины в противоположную сторону (как правило, этот параметр зависит от величины пропорционального коэффициента k_p).

Напишите программу на языке программирования Python, реализующую ПИД или любой из его вариаций регулятор высоты $H^{\text{зад}}$ и регулятор приборной скорости $V_{\text{пр}}^{\text{зад}}$. Характеристики переходных процессов заданных величин должны находиться в рамках требований к переходным процессам.

БВС осуществляет полёт в самолётном режиме на высоте H_0(м), с приборной скоростью $V_{\text{пр0}}$(км/ч), углы поворота ВМГ $\varepsilon_1, \varepsilon_2 = 90°$ зафиксированы, значения отклонений элеронов первого крыла $\delta_{\text{эл1}} = \delta_{\text{эл}}^{11} = \delta_{\text{эл}}^{12} = 0°$.

Шаблоном программы является класс SAU, включённый в моделирования посредством классов-интерфейсов SAU_in и SAU_out.

Код и структура класса SAU_in (на языке Python) имеют вид:

```
class SAU_in:
    # Текущий угол наклона траектории, град.
    Tet = 0
    # Текущий угол тангажа, град.
    Tan = 0
    # Текущая угловая скорость, связанная СК, град/сек
    Wz1 = 0
    # Текущая высота, м.
    H = 0
```

```
    # Текущая вертикальная скорость,м/с
    Vy = 0
    # Текущая приборная скорость, км/ч
    Vw = 0
```

Код и структура класса SAU_out (на языке Python) имеют вид:

```
class SAU_out:
    # Заданные значения положения дросселя для ВМГ 1,2,  от 0.1
до 1
    P12_dr = 0.2
    # Заданные значения положения дросселя для ВМГ 3,4, от 0.1
до 1
    P34_dr = 0.2
    # Заданное значение угла поворота ВМГ первого и второго
крыла, град
    eps = 90.0
    # Заданное значения угла поворота элеронов второго крыла
    de = 0.0
```

Код и структура класса SAU (на языке Python) имеют вид:

```
# Функция ограничения значения
clamp = lambda n, minn, maxn: max(min(maxn, n), minn)

class SAU:
    def __init__(self,H_zad,Vpr_zad,eps,L_zad):
        self.u_output = SAU_out()
        self.u_output.eps = eps
        # Шаг интегрирования
        self.dt = 0.01
        # Время моделирования
        self.Time = 0
        # Заданный угол тангажа
        self.TangU = 0
        # Производная заданного угла тангажа
        self.TangUDt = 0
        # Интеграл ошибки
        self.Integral_H = 0
        # Заданное значение положения дросселя для ВМГ 1,2
        self.P_reg_12 = 0
        # Заданное значение положения дросселя для ВМГ 3,4
        self.P_reg_34 = 0
        # Заданный угол отклонения элеронов второго крыла
sigma_3,sigma_4
        self.delta_elerons = 0

        # Заданные регулируемые значения
        # Заданная высота, м.
        self.H_zad = H_zad
        # Заданная приборная скорость, км/ч
        self.Vpr_zad = Vpr_zad
        # Заданное значение
```

```
        self.L_zad = L_zad

        # Плечи
        self.lz1 = 1
        self.lz2 = 0.84
        self.ly1 = 0.036
        self.ly2 = 0.14

        # Масса БВС
        self.m = 108

        # Для получения дроссельных характеристик в зависимости
от расчетной тяги

        self.drossel = np.linspace(0, 1, 10, endpoint=True) #
Положение дросселя, 0..1
        self.P_one = np.linspace(0, 350, 10, endpoint=True) #
Тяга,Ньютон

        # Шаблон записи для отладки
        self.writenames
=list(["Time","TanU","TanU_dt","P_reg_12","P_reg_34"])
        data_sau =
[self.Time,self.TangU,self.TangUDt,self.P_reg_12,self.P_reg_34]

        self.db = DataFrame([data_sau],columns =
self.writenames)
    # Функция записи данных
    def writeframe(self):
        data_sau =
[self.Time,self.TangU,self.TangUDt,self.P_reg_12,self.P_reg_34]
        frame = DataFrame([data_sau],columns = self.writenames)
        self.db = concat([self.db, frame], ignore_index=True)

    # Шаблонная функция вычисления заданного положения дросселя
для регулирования приборной скорости
    def calc_drossel(self):
        # Заданные значения
        # Заданная приборная скорость, км/ч
        Vpr_zad = self.Vpr_zad/3.6
        # Параметры которые могут понадобится для расчета
        # Текущая приборная скорость, км/ч
        Vpr_now =  self.u_input.Vw/3.6


        self.P_reg = 0.1
        # Ограничение [0.1,1] обязательно
        self.drossel = clamp(self.P_reg,0.1,1)
        self.P_reg_12  =  self.drossel*0.5
        self.P_reg_34  =  self.drossel*0.5
```

```
    # Шаблонная функция вычисления заданного отклонения
элеронов второго крыла для регулирования высоты
    def calc_delta_elerons(self):
        # Заданные значения
        H_zad = self.H_zad
        # Параметры которые могут понадобится для расчета
        # Текущий угол тангажа, град.
        Tang = self.u_input.Tan
        # Текущая угловая скорость по оси OZ, в связанной СК
        Wz1 = self.u_input.Wz1
        # Текущая высота БВС, м.
        H_now = self.u_input.H
        # Текущая вертикальная скорость БВС, м.
        Vy = self.u_input.Vy

        delta_elerons_calc =0

        # Ограничение -25 +25 обязательно
        self.delta_elerons = clamp(delta_elerons_calc,-25,25)

    # Основная функция расчета, вызывается при моделировании
    def calc_PID(self,input:SAU_in):

        self.u_input = input
        # Вызов шаблонной функции, расчета дросселя
        self.calc_drossel()
        # Вызов шаблонной функции, расчета элеронов заднего
крыла
        self.calc_delta_elerons()
        # Запись параметров на мат. модель
        self.u_output.de = self.delta_elerons  # Заданное
значение положения элеронов второго крыла
        self.u_output.P12_dr = self.P_reg_12    # Заданное
значение дросселя ВМГ 1-2
        self.u_output.P34_dr = self.P_reg_34    # Заданное
значение дросселя ВМГ 1-2
        self.Time += self.dt
        self.writeframe()
        return self.u_output

    def drop_u(self):
        return self.u_output
```

Рекомендуется производить вычисления в функциях-шаблонах:

- calc_delta_elerons – шаблонная функция вычисления заданного отклонения элеронов второго крыла для регулирования высоты;
- calc_drossel – шаблонная функция вычисления заданного положения дросселя для регулирования приборной скорости.

Участники олимпиады также могут выполнить проверку своего решения по соответствию требованиям к переходным процессам при помощи функции step_info –

функции проверки соответствия характеристик процесса регулирования требованиям к процессу регулирования.

Условия моделирования полёта БВС:

- начальное значение угла тангажа Tang = 1,5 град.;
- начальное значение угла наклона траектории Theta = 0 град.;
- начальное значение угла атаки Alpha = 1,5 град.;
- начальное значение земной скорости Va = 55,5 м/с;
- начальное значение приборной скорости Vpr = 200 км/ч;
- начальное значение высоты БВС H = 1000 м.

Заданные параметры полета БВС:

- заданное значение высоты БВС H = 800 м;
- заданное значение приборной скорости БВС Vpr = 200 км/ч.

Требования к переходному процессу по высоте H и приборной скорости Vpr:

	Высокая точность	Средняя точность	Низкая точность
Статическая ошибка	$\Delta X_{\text{треб}} \le 2{,}3$	$2{,}3 < \Delta X_{\text{треб}} \le 2{,}7$	$\Delta X_{\text{треб}} > 2{,}7$
Время переходного процесса	$t_{\text{пер}} \le 17{,}25$	$17{,}25 < t_{\text{пер}} \le 20{,}25$	$t_{\text{пер}} > 20{,}25$
Перерегулирование	$\Delta X \le 11{,}5$	$11{,}5 < \Delta X \le 13{,}5$	$\Delta X > 13{,}5$

Решение:

```
from UAV import SAU_in,SAU_out
from math import *
from pandas import DataFrame, concat
import numpy as np
# Функция ограничения значения
clamp = lambda n, minn, maxn: max(min(maxn, n), minn)

class SAU:
    def __init__(self,H_zad,Vpr_zad,eps,L_zad):
        self.u_output = SAU_out()
        self.u_output.eps = eps
        # Шаг интегрирования
        self.dt = 0.01
        # Время моделирования
        self.Time = 0
        # Заданный угол тангажа
        self.TangU = 0
        # Производная заданного угла тангажа
        self.TangUDt = 0
        # Интеграл ошибки
```

```
        self.Integral_H = 0
        # Заданное значение положения дросселя для ВМГ 1,2
        self.P_reg_12 = 0
        # Заданное значение положения дросселя для ВМГ 3,4
        self.P_reg_34 = 0
        # Заданный угол отклонения элеронов второго крыла
sigma_3,sigma_4
        self.delta_elerons = 0

        # Заданные регулируемые значения
        # Заданная высота, м.
        self.H_zad = H_zad
        # Заданная приборная скорость, км/ч
        self.Vpr_zad = Vpr_zad
        # Заданное значение
        self.L_zad = L_zad

        # Плечи
        self.lz1 = 1
        self.lz2 = 0.84
        self.ly1 = 0.036
        self.ly2 = 0.14

        # Масса БВС
        self.m = 108

        # Для получения дроссельных характеристик в зависимости
от расчетной тяги

        self.drossel = np.linspace(0, 1, 10, endpoint=True) #
Положение дросселя, 0..1
        self.P_one = np.linspace(0, 350, 10, endpoint=True) #
Тяга,Ньютон

        # Шаблон записи для отладки
        self.writenames
=list(["Time","TanU","TanU_dt","P_reg_12","P_reg_34"])
        data_sau =
[self.Time,self.TangU,self.TangUDt,self.P_reg_12,self.P_reg_34]

        self.db = DataFrame([data_sau],columns =
self.writenames)
    # Функция записи данных
    def writeframe(self):
        data_sau =
[self.Time,self.TangU,self.TangUDt,self.P_reg_12,self.P_reg_34]
        frame = DataFrame([data_sau],columns = self.writenames)
        self.db = concat([self.db, frame], ignore_index=True)
```

```
    # Шаблонная функция вычисления заданного положения дросселя
для регулирования приборной скорости
    def calc_drossel(self):
        # Заданные значения
        # Заданная приборная скорость, км/ч
        Vpr_zad = self.Vpr_zad/3.6
        # Параметры которые могут понадобится для расчета
        # Текущая приборная скорость, км/ч
        Vpr_now =  self.u_input.Vw/3.6
        # Расчет
        k_P = 0.5 # Пропорциональный коэффициент
        self.P_reg = 0.1 + k_P*(Vpr_zad - Vpr_now)
        # Ограничение [0.1,1] обязательно
        self.drossel = clamp(self.P_reg,0.1,1)
        self.P_reg_12  =  self.drossel*0.5
        self.P_reg_34  =  self.drossel*0.5

    # Шаблонная функция вычисления заданного отклонения
элеронов второго крыла для регулирования высоты
    def calc_delta_elerons(self):
        # Заданные значения
        H_zad = self.H_zad
        # Параметры которые могут понадобится для расчета
        # Текущий угол тангажа, град.
        Tang = self.u_input.Tan
        # Текущая угловая скорость по оси OZ, в связанной СК
        Wz1 = self.u_input.Wz1
        # Текущая высота БВС, м.
        H_now = self.u_input.H
        # Текущая вертикальная скорость БВС, м.
        Vy = self.u_input.Vy
        # Эталонный расчет

        # Пропорциональное звено
        k_P = 7.5
        # Интегральное звено
        k_i = 0.1
        # Дифференциальное звено
        k_d = 1.0

        # Заданный угол тангажа в зависимости от высоты, с
ограничением
        self.TangU = clamp(-1.0 * (H_now - H_zad) - 0.5 * Vy,-
20,20)
        delta_elerons_calc = k_P*( self.TangU  - Tang ) - k_d *
Wz1 - k_i * self.Integral_H
        # Интеграл ошибки
        self.Integral_H += (H_now-H_zad)*self.dt
        # Ограничение -25 +25 обязательно
        self.delta_elerons = clamp(delta_elerons_calc,-25,25)

    # Основная функция расчета, вызывается при моделировании
```

```
    def calc_PID(self,input:SAU_in):

        self.u_input = input
        # Вызов шаблонной функции, расчета дросселя
        self.calc_drossel()
        # Вызов шаблонной функции, расчета элеронов заднего
крыла
        self.calc_delta_elerons()
        # Запись параметров на мат. модель
        self.u_output.de = self.delta_elerons  # Заданное
значение положения элеронов второго крыла
        self.u_output.P12_dr = self.P_reg_12    # Заданное
значение дросселя ВМГ 1-2
        self.u_output.P34_dr = self.P_reg_34    # Заданное
значение дросселя ВМГ 1-2
        self.Time += self.dt
        self.writeframe()
        return self.u_output

    def drop_u(self):
        return self.u_output
```