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

Категория участия «Магистратура/специалитет»

Вариант № 1

Задача № 1

Беспилотное воздушное судно (БВС) самолётного типа с толкающим воздушным винтом совершает горизонтальный неускоренный полёт на крейсерской высоте 3 км. Рассчитайте диаметр воздушного винта и определите, каким может быть максимальное лобовое сопротивление этого БВС, если винт подобран для обеспечения указанного режима полёта. Округлите ответ до десятых.

Дополнительные данные:

- коэффициент мощности воздушного винта $C_p = \frac{P}{\rho n^3 D^5} = 0{,}06$;
- коэффициент тяги воздушного винта $C_T = \frac{T}{\rho n^2 D^4} = 0{,}1$;
- P – располагаемая мощность воздушного винта на крейсерском режиме полета самолёта, [Вт], $P = 37$ кВт;
- T – тяга воздушного винта, [Н];
- ρ – плотность воздуха [кг/м^3]; на высоте крейсерского полёта данного самолёта (3 км) плотность воздуха равна 0,9 кг/м^3;
- n – частота вращения воздушного винта, [об/с]; на крейсерском режиме полёта самолёта частота вращения данного винта 2400 об/мин.;
- D – диаметр воздушного винта, [м].

Решение (основные вычисления):

$$C_p = \frac{P}{\rho n^3 D^5}$$

$$D = \sqrt[5]{\frac{P}{C_p \rho n^3}} = \sqrt[5]{\frac{37000}{0{,}06 * 0{,}9 * \left(\frac{2400}{60}\right)^3}} = 1{,}6\text{м}$$

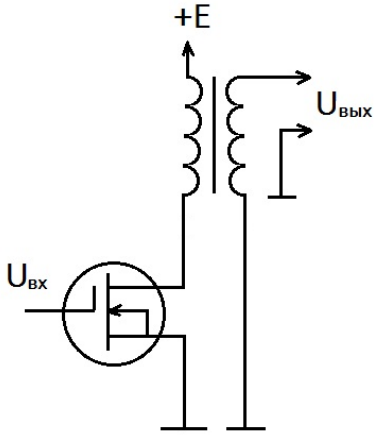
Лобовое сопротивление в горизонтальном неускоренном полете равно тяге винта:

$X = T = C_T \rho n^2 D^4 = 0{,}1 * 0{,}9 * \left(\frac{2400}{60}\right)^2 = 1{,}6^4 = 959{,}5$ Н.

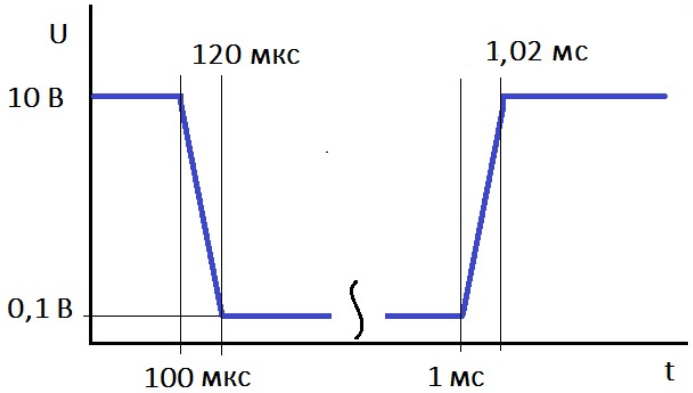
Ответ: диаметр воздушного винта 1,6 м, максимальное лобовое сопротивление 959,5 Н.

Задача № 2

Беспилотное воздушное судно (БВС) самолётного типа имеет в качестве силовой установки поршневой двигатель с электронной системой зажигания. Определить напряжение искры на вторичной обмотке катушки зажигания, если индуктивность первичной обмотки составляет 10 мГн при 1000 витках, а количество витков вторичной обмотки 20 000.



Напряжение на стоке транзистора показано на графике, сопротивление канала транзистора в открытом состоянии принять равным 100 мОм. Напряжение питания E = 10 В. Ответ дать с точностью до целых.



Решение (основные вычисления):

Из графика видно, что величина тока стока изменяется от $I = \frac{U}{R} = \frac{0{,}1}{100 \cdot 10^{-3}} = 1$ А до нуля за время, равное 20 мкс. Тогда величина перепада напряжения на первичной обмотке составит: $U_1 = \frac{LdI}{dt} = \frac{0{,}01 \cdot 1}{20 \cdot 10^{-6}} = 500$ В.

С учётом коэффициента трансформации, напряжение на вторичной обмотке составит: $U_2 = U_1 \frac{N_2}{N_1} = 500 \cdot \frac{20000}{1000} = 10\,000$ В $= 10$ кВ

Ответ: 10 кВ.

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).

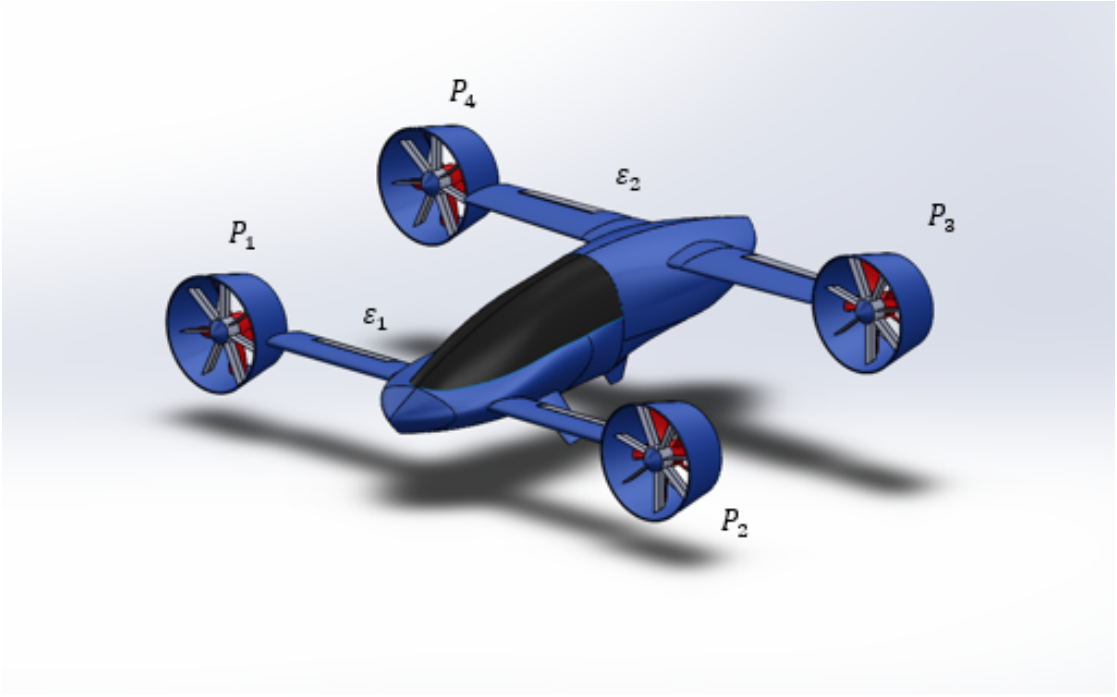


Рисунок 1. Внешний облик БВС, конвертоплан схемы тандем.

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.

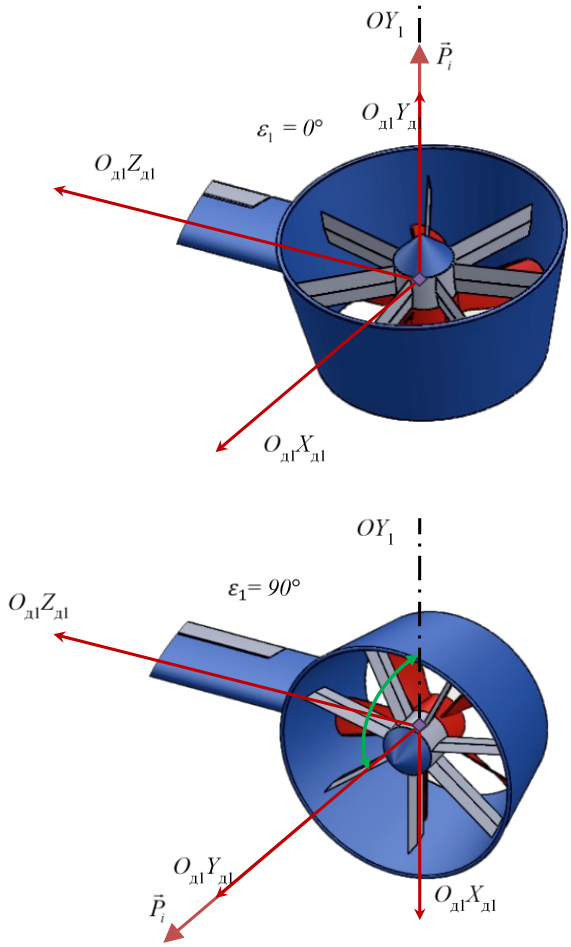


Рисунок 2. Вращение ВМГ вокруг оси ε_j.

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.

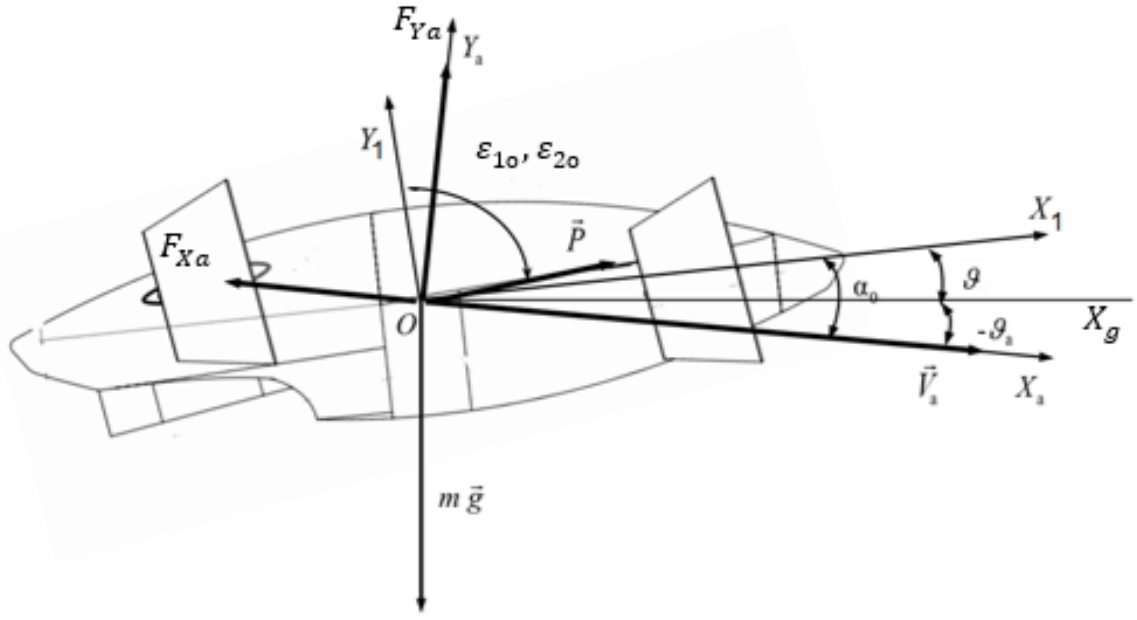


Рисунок 3. Силы, действующие на БВС в самолётном режиме.

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:

$$G = m \cdot g.$$

Помимо линейных перемещений на БВС также имеет угловое движение вокруг центра масс в продольном канале на которое влияют аэродинамический момент M_z^a и момент от ВМГ M_z^P.

Значение аэродинамического момента M_z^a находится при помощи выражения:

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

Суммарный момент M_z^P, создаваемый ВМГ, зависит от тяги пар ВМГ расположенных на переднем крыле P_1, P_2 и на заднем крыле P_3, P_4, а также зависит от угла поворота пар ВМГ $\varepsilon_1, \varepsilon_2$.

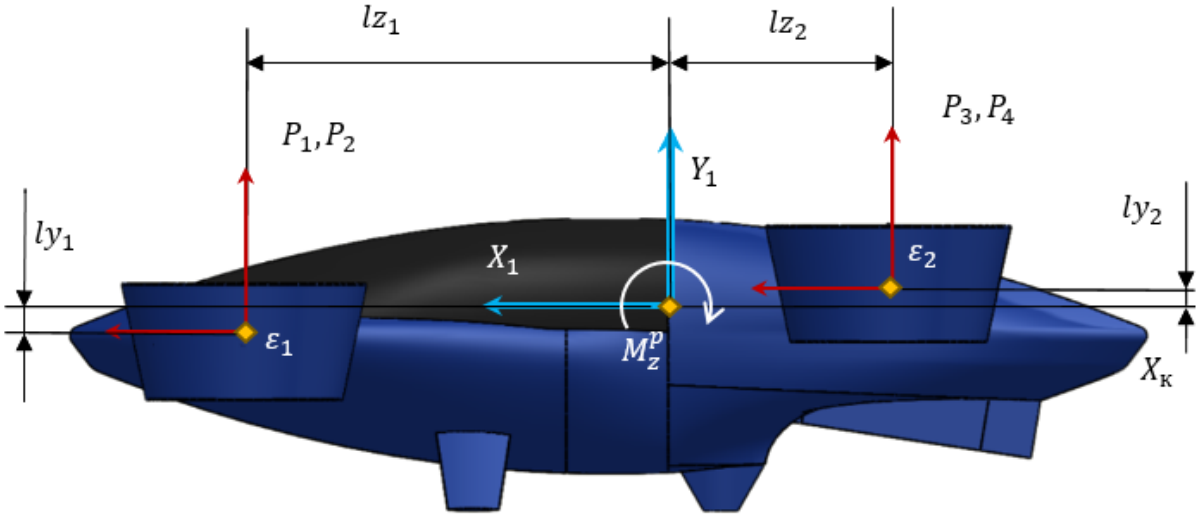


Рисунок 4. Плечи сил тяги ВМГ, момент силы тяги M_z^P.

Стоит отметить, что в продольном движении можно считать, что угол тангажа ϑ вычисляется исходя из следующего соотношения:

$$\vartheta = \theta + \alpha.$$

Конвертоплан совершает установившийся горизонтальный прямолинейный полёт (V = const, $\theta = 0$, H = const). Определите величину углов отклонения оси винтомоторной группы $\varepsilon_1, \varepsilon_2$, а так же величины тяг ВМГ P_1, P_2, P_3, P_4 при условии что $\varepsilon_1 = \varepsilon_2$. Ответ указать в градусах и Н, с точностью сотых.

При расчетах 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;
- характерный линейный размер, $l = 0.3$ м;
- плечо силы $lz_1 = 1{,}0$ м;
- плечо силы $lz_2 = 0{,}8$ м;
- плечо силы $ly_1 = 0{,}036$ м;
- плечо силы $ly_2 = 0{,}14$ м;
- воздушная скорость $V_a = 40$ м/с;
- угол наклона траектории $\theta = 0°$;
- угол атаки $\alpha = 1{,}6°$;
- коэффициент лобового сопротивления $c_x(\alpha) = 0{,}0626$;
- коэффициент подъемной силы $c_y(\alpha) = 0{,}1631$;
- коэффициент аэродинамического момента $m_z(\alpha) = -0{,}1341$.

Решение (основные вычисления):

Момент от ВМГ по оси Z (гироскопическим моментом пренебрегаем): $M_z^P = M_z^{lz} + M_z^{ly}$

С учётом того, что $P_1 = P_2$ а $P_3 = P_4$,

$$M_z^{lz} = 2 \cdot cos(\varepsilon) \cdot (P_1 \cdot lz_1 - P_3 \cdot lz_2)$$
$$M_z^{ly} = 2 \cdot sin(\varepsilon) \cdot (P_1 \cdot ly_1 - P_3 \cdot ly_2)$$

Система 1:

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

Система 2:

$$2 \cdot (P_1 + P_3) = P$$
$$M_z^P + M_z^a = 0$$

Исходя из системы 1:

$$\varepsilon = -\operatorname{atan}\left(\frac{m \cdot g - c_y(\alpha)}{c_x(\alpha)}\right) \cdot \left(\frac{180°}{\pi}\right) - 90° - \alpha;$$
$$\sum_{i=1}^{4} P = \frac{c_x(\alpha)\frac{\rho V^2}{2}S}{\cos(90° + \alpha - \varepsilon)};$$

Исходя из системы 2:

$$\sum_{i=1}^{2} P = \frac{P \cdot (ly_2 \cdot sin(\varepsilon) + lz_2 \cdot cos(\varepsilon)) - M_z^a}{cos(\varepsilon) \cdot (lz_1 + lz_2) + sin(\varepsilon) \cdot (ly_1 + ly_2)};$$
$$\sum_{i=3}^{4} P = P - \sum_{i=1}^{2} P$$

Тяга для каждого ВМГ:

$$P_1 = P_2 = \sum_{i=1}^{2} P \Big/ 2$$

$$P_3 = P_4 = \sum_{i=3}^{4} P \Big/ 2$$

Ответ: $\varepsilon_1 = \varepsilon_2 = 12.12°$; $P_1, P_2 = 187.64$ Н; $P_3, P_4 = 175.34$ 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 БВС находится в вертолётном режиме, необходимо написать программы: систему стабилизации и систему траекторного управления для продольного канала движения БВС в указанные координаты (высота, дальность).

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.

По умолчанию БВС сбалансирован на начальной высоте.

В рамках данной задачи для осуществления стабилизации и траекторного управления БВС необходимо будет построить соответствующие ПИД-регуляторы.

Система стабилизации:

- угол тангажа $\vartheta - M_z^u$;
- высоты $H - P_y^u$.

Система траекторного управления:

- продольная скорость $V_{xg} - V_x^u$.

Структура автоматического управления БВС типа конвертоплан в вертолётном режиме для продольного движения показана рисунке 1.

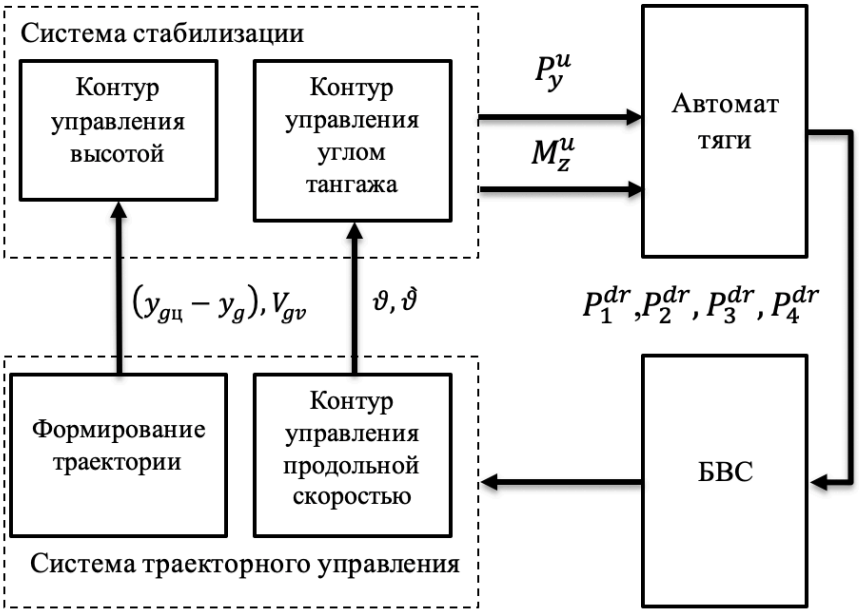


Рисунок 1. Структура САУ продольным движением БВС в вертолётном режиме управления.

Реализовать алгоритм возможно при помощи пропорционально-интегрально-дифференциального регулятора (ПИД-регулятора), структурная схема котрого показана на рисунке 2.

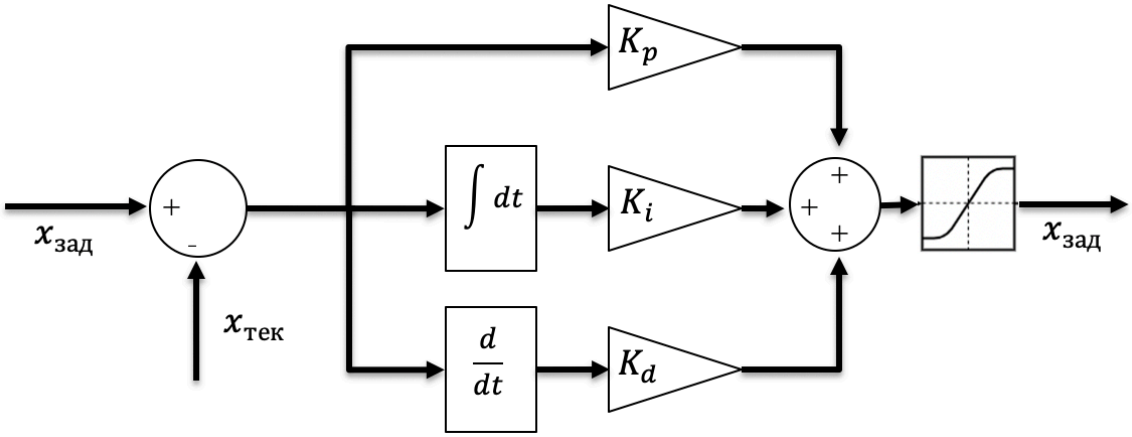


Рисунок 2. Структурная схема ПИД-регулятора.

Как показано на рисунке 2, 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 3:

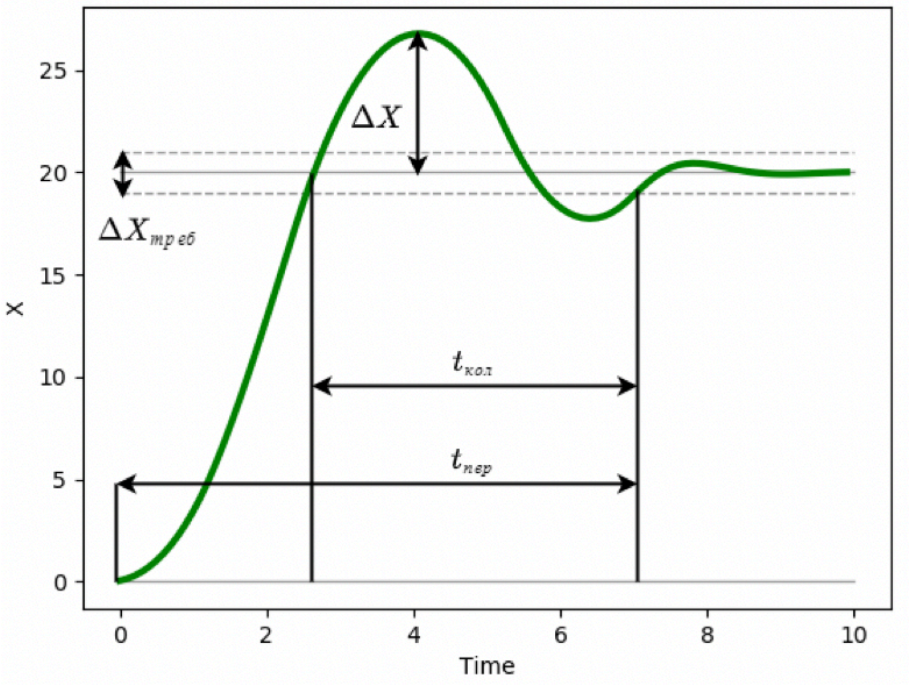


Рисунок 3. Переходный процесс регулируемой величины.

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).

Задачей траекторного управления является перемещение БВС из его текущего местоположения, определяющего отклонение БВС от линии пути А-Б, в точку Б путём сокращения продольного $l_{\text{цп}}$. Так как в задаче рассматривается только продольное движение, угол курса $\psi = \psi_{\text{пот}} = const = 0$, соответственно дальность до точки Б $L_{\text{ц}} = l_{\text{цп}}$ – сокращение $l_{\text{цп}}$ производится за счёт изменения угла тангажа ϑ и последующего появления продольной компоненты линейной скорости V_{xg} (рисунок 4).

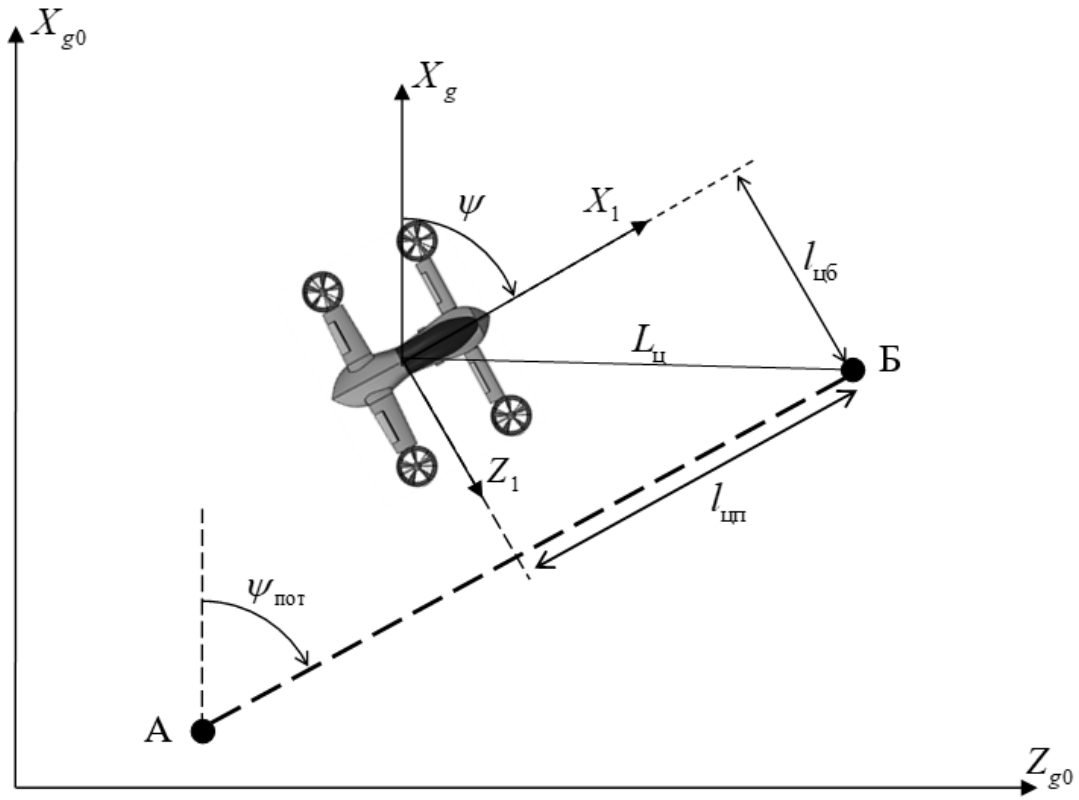


Рисунок 4. Параметры характеризующие расположения БВС относительно линии пути А–Б.

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

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

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
```

Участникам олимпиады рекомендуется производить вычисления в функциях-шаблонах:

- calc_stab – шаблонная функция исполнения системы стабилизации;
- calc_traj – шаблонная функция исполнения системы траекторного управления.

Условия моделирования полёта БВС:

- начальное значение угла тангажа Tang = 0 град.;
- начальное значение угла наклона траектории Theta = 0 град.;
- начальное значение приборной скорости Vpr = 0 км/ч;
- начальное значение высоты БВС H = 10 м.

Заданные параметры полёта БВС:

- заданное дальность до точки L_zad = 50 м;

- заданное высота точки H_zad = 50 м.

Требования к переходному процессу заданного угла тангажа:

	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	$t_{пер} \le 23$	$23 < t_{пер} \le 27$	$t_{пер} > 27$
Перерегулирование	$\Delta X \le 23$	$23 < \Delta X \le 27$	$\Delta X > 27$

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

        self.Vy_prev = 0
        self.VyI = 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 и высоты
     # По умолчанию стабилизирует БВС на заданной высоте.
    def stab_process(self):
        # Заданные значения
        H_zad = self.H_zad
        # Параметры которые могут понадобится для расчета
        # Текущая приборная скорость (можно считать что
горизонтальная), м/с
        Vx = self.u_input.Vw/3.6
        # Текущий угол тангажа, град.
        Tang = self.u_input.Tan
```

```
        # Текущая угловая скорость по оси OZ, в связанной СК,
град.
        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
        # Текущее положение угла поворота ВМГ, рад.
        eps = radians(self.u_output.eps)
        # Плечи
        lz1 = self.lz1
        lz2 = self.lz2

        ly1 = self.ly1
        ly2 = self.ly2

        sinE = sin(eps)
        cosE = cos(eps)
        L_p =   self.L_zad - self.u_input.L
        #
        P_trim_sum =  self.m*9.87/cos(Tang)
        # Требуемая тяга ДВС для поддержания БВС в состоянии
стабилизации на текущей высоте.

        Y_ksu = H_now - H_zad
        Vy_tr = - Y_ksu  / max(3,L_p)
        Vy_zad = clamp(180*Vy_tr,-7,7)
        ay = (Vy - self.Vy_prev)/self.dt
        self.Vy_prev = Vy

        self.VyI =  self.VyI + (Vy - self.Vy_prev) * self.dt

        R = P_trim_sum -  0*self.VyI - 10*( Vy - Vy_zad) +
60.5*ay

        k12 = lz2/(2*(lz1 + lz2))
        k34 = lz1/(2*(lz1 + lz2))

        P12_zad = (k12)*(R)
        P34_zad = (k34)*(R)

        dr12 = np.interp(P12_zad,self.P_one,self.drossel)
        dr34 = np.interp(P34_zad,self.P_one,self.drossel)

        self.P_reg_12 =  dr12
        self.P_reg_34 =  dr34


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

        self.u_input = input
```

```
        # Вызов шаблонной функции
        self.stab_process()
        # 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

Беспилотное воздушное судно (БВС) самолётного типа с толкающим воздушным винтом совершает горизонтальный неускоренный полёт на крейсерской высоте 3 км. Рассчитайте диаметр воздушного винта и определите, каким может быть максимальное лобовое сопротивление этого БВС, если винт подобран для обеспечения указанного режима полёта. Округлите ответ до десятых.

Дополнительные данные:

- коэффициент мощности воздушного винта $C_p = \frac{P}{\rho n^3 D^5} = 0{,}06$;
- коэффициент тяги воздушного винта $C_T = \frac{T}{\rho n^2 D^4} = 0{,}1$;
- P – располагаемая мощность воздушного винта на крейсерском режиме полета самолёта, [Вт], P = 39 кВт;
- T – тяга воздушного винта, [Н];
- ρ – плотность воздуха [кг/м^3]; на высоте крейсерского полёта данного самолёта (3 км) плотность воздуха равна 0,9 кг/м^3;
- n – частота вращения воздушного винта, [об/с]; на крейсерском режиме полёта самолёта частота вращения данного винта 2300 об/мин.;
- D – диаметр воздушного винта, [м].

Решение (основные вычисления):

$$C_p = \frac{P}{\rho n^3 D^5}$$

$$D = \sqrt[5]{\frac{P}{C_p \rho n^3}} = \sqrt[5]{\frac{39000}{0{,}06 * 0{,}9 * \left(\frac{2300}{60}\right)^3}} = 1{,}7\text{м}$$

Лобовое сопротивление в горизонтальном неускоренном полёте равно тяге винта

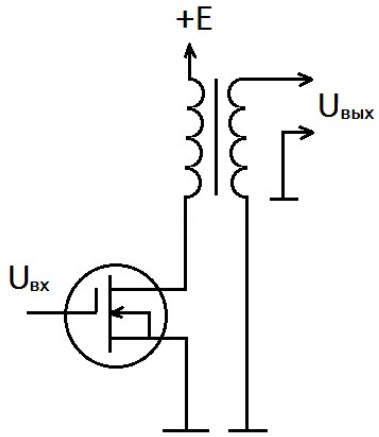
$$X = T = C_T \rho n^2 D^4 = 0{,}1 * 0{,}9 * \left(\frac{2300}{60}\right)^2 = 1{,}6^4 = 1018\text{Н}$$

Ответ: диаметр воздушного винта 1,7 м, максимальное лобовое сопротивление 1018,0 Н.

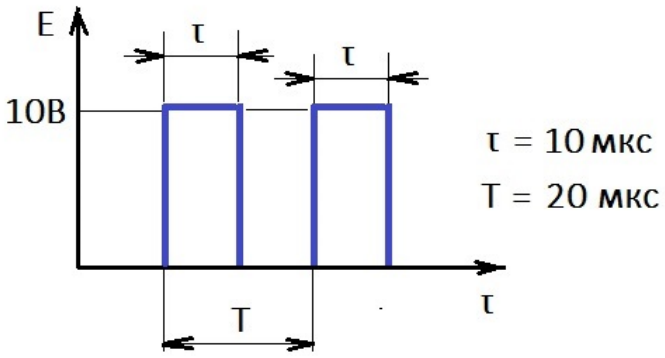
Задача № 2

Беспилотное воздушное судно (БВС) самолётного типа имеет в качестве силовой установки поршневой двигатель с электронной системой зажигания, в которой применяется RC-цепь интегрирующего типа.

На вход RC-цепи, изображенной на рисунке,



подаётся импульсная последовательность, показанная на рисунке



Определить максимальное напряжение на конденсаторе. Ответ дать с точностью до тысячных.

Решение (основные вычисления):

Можно рассматривать импульсы как отдельные воздействия (принцип суперпозиции), тогда искомое напряжение будет определено как сумма воздействий от обоих импульсов.

Напряжение в момент завершения первого импульса можно определить, как $U_1 = E\left(1 - e^{-\frac{\tau}{RC}}\right) = 10\left(1 - e^{-\frac{10}{1000 \cdot 0{,}01}}\right) = 6{,}321$ В.

Далее конденсатор будет разряжаться и к моменту завершения второго импульса напряжение на нём будет равно

$U'_1 = U_1\left(e^{-\frac{\tau \cdot 2}{RC}}\right) = 6{,}321\left(e^{-\frac{10 \cdot 2}{1000 \cdot 0{,}01}}\right) = 0{,}855$ В.

Напряжение на конденсаторе, очевидно, будет максимально к моменту завершения второго импульса, и равно $U_{max} = U'_1 + U_2$, где U_2 будет равно U_1, так как импульсы идентичны — $U_2 = U_1$.

В итоге имеем $U_{max} = U'_1 + U_1 = 0{,}855 + 6{,}321 = 7{,}176$ В

Ответ: 7,176 В.

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).

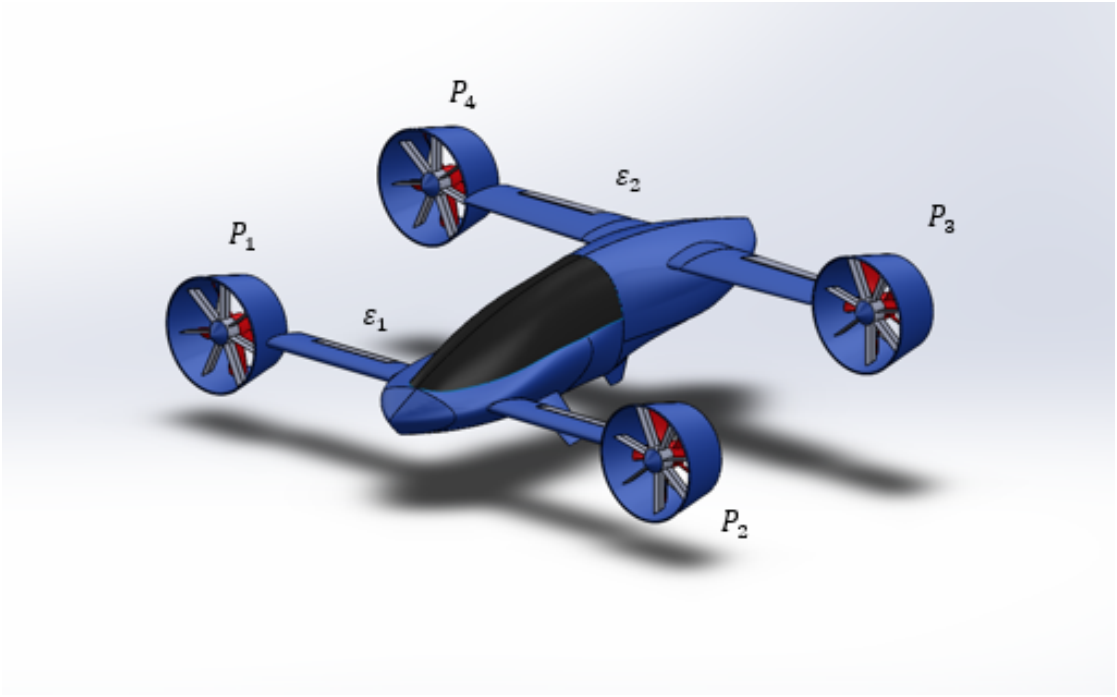


Рисунок 1. Внешний облик БВС, конвертоплан схемы тандем.

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.

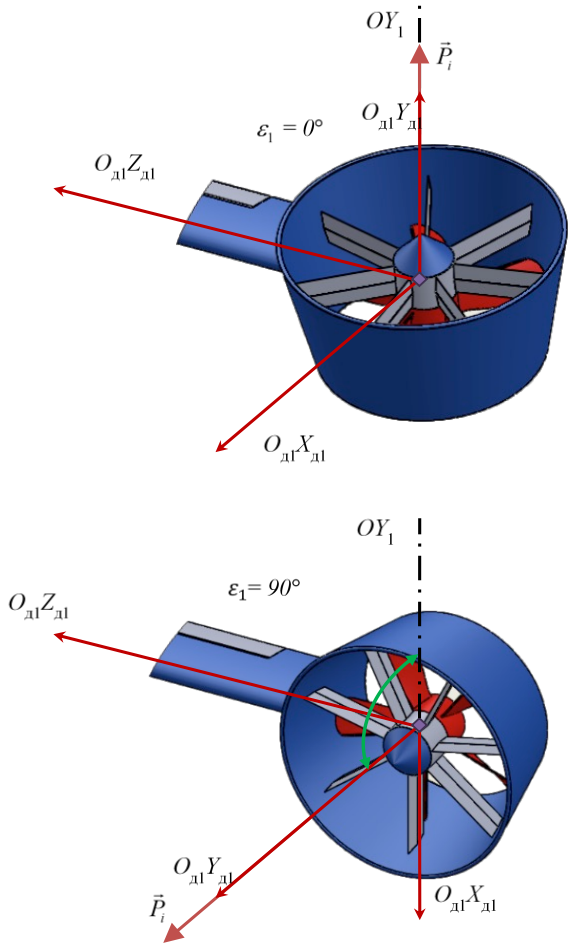


Рисунок 2. Вращение ВМГ вокруг оси ε_j.

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.

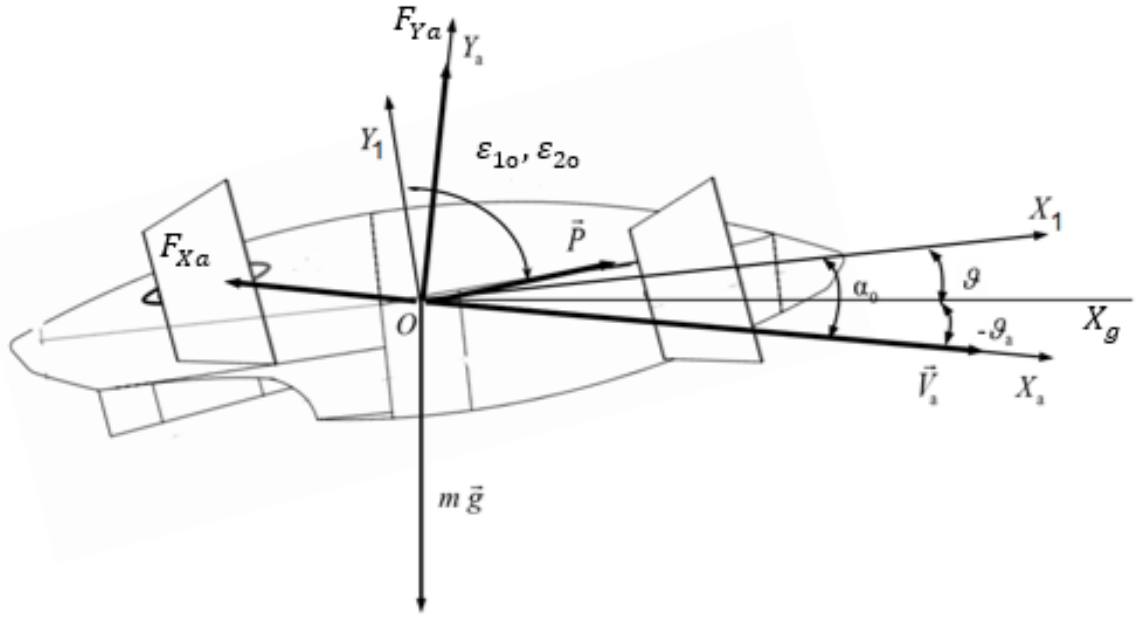


Рисунок 3. Силы, действующие на БВС в самолётном режиме.

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:

$$G = m \cdot g.$$

Помимо линейных перемещений на БВС также имеет угловое движение вокруг центра масс в продольном канале на которое влияют аэродинамический момент M_z^a и момент от ВМГ M_z^P.

Значение аэродинамического момента M_z^a находится при помощи выражения:

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

Суммарный момент M_z^P, создаваемый ВМГ, зависит от тяги пар ВМГ расположенных на переднем крыле P_1, P_2 и на заднем крыле P_3, P_4, а также зависит от угла поворота пар ВМГ $\varepsilon_1, \varepsilon_2$.

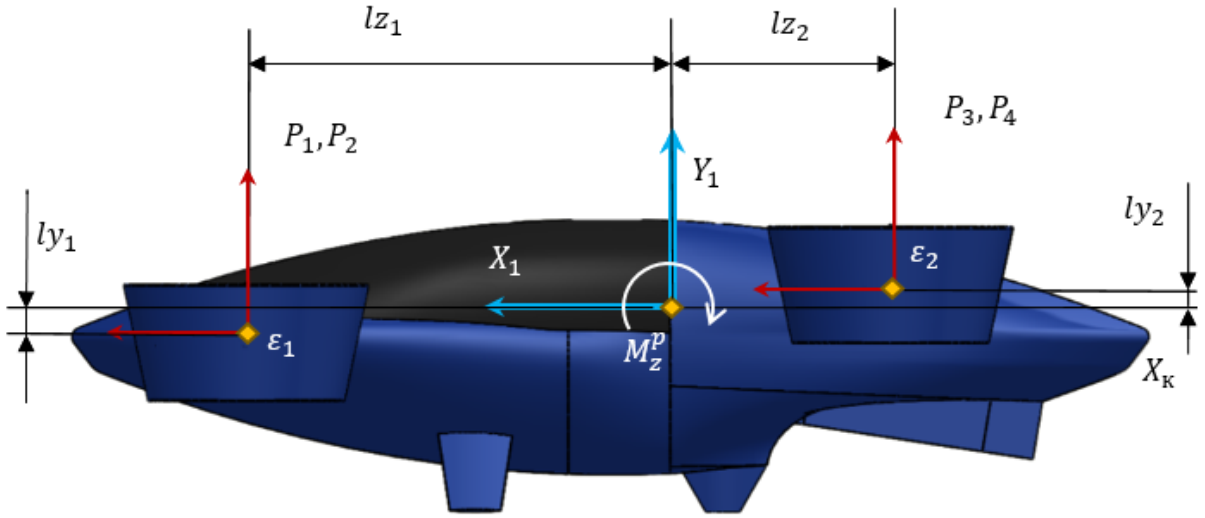


Рисунок 4. Плечи сил тяги ВМГ, момент силы тяги M_z^P.

Стоит отметить, что в продольном движении можно считать, что угол тангажа ϑ вычисляется исходя из следующего соотношения:

$$\vartheta = \theta + \alpha.$$

Конвертоплан совершает установившийся горизонтальный прямолинейный полёт (V = const, $\theta = 0$, H = const). Определите величину углов отклонения оси винтомоторной группы $\varepsilon_1, \varepsilon_2$, а так же величины тяг ВМГ P_1, P_2, P_3, P_4 при условии что $\varepsilon_1 = \varepsilon_2$. Ответ указать в градусах и Н, с точностью сотых.

При расчетах 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;

- характерный линейный размер, $l = 0.3$ м;
- плечо силы $lz_1 = 1{,}0$ м;
- плечо силы $lz_2 = 0{,}8$ м;
- плечо силы $ly_1 = 0{,}036$ м;
- плечо силы $ly_2 = 0{,}14$ м;
- воздушная скорость $V_a = 50$ м/с;
- угол наклона траектории $\theta = 0°$;
- угол атаки $\alpha = 2{,}3°$;
- коэффициент лобового сопротивления $c_x(\alpha) = 0{,}0669$;
- коэффициент подъемной силы $c_y(\alpha) = 0{,}2032$;
- коэффициент аэродинамического момента $m_z(\alpha) = -0{,}2443$.

Решение (основные вычисления):

Момент от ВМГ по оси Z (гироскопическим моментом пренебрегаем): $M_z^P = M_z^{lz} + M_z^{ly}$

С учётом того, что $P_1 = P_2$ а $P_3 = P_4$,

$$M_z^{lz} = 2 \cdot cos(\varepsilon) \cdot (P_1 \cdot lz_1 - P_3 \cdot lz_2)$$
$$M_z^{ly} = 2 \cdot sin(\varepsilon) \cdot (P_1 \cdot ly_1 - P_3 \cdot ly_2)$$

Система 1:

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

Система 2:

$$2 \cdot (P_1 + P_3) = P$$
$$M_z^P + M_z^a = 0$$

Исходя из системы 1:

$$\varepsilon = -\operatorname{atan}\left(\frac{m \cdot g - c_y(\alpha)}{c_x(\alpha)}\right) \cdot \left(\frac{180°}{\pi}\right) - 90° - \alpha;$$
$$\sum_{i=1}^{4} P = \frac{c_x(\alpha)\frac{\rho V^2}{2} S}{\cos(90° + \alpha - \varepsilon)};$$

Исходя из системы 2:

$$\sum_{i=1}^{2} P = \frac{P \cdot (ly_2 \cdot sin(\varepsilon) + lz_2 \cdot cos(\varepsilon)) - M_z^a}{cos(\varepsilon) \cdot (lz_1 + lz_2) + sin(\varepsilon) \cdot (ly_1 + ly_2)};$$
$$\sum_{i=3}^{4} P = P - \sum_{i=1}^{2} P$$

Тяга для каждого ВМГ:

$$P_1 = P_2 = \sum_{i=1}^{2} P \Big/ 2$$

$$P_3 = P_4 = \sum_{i=3}^{4} P \Big/ 2$$

Ответ: $\boldsymbol{\varepsilon_1 = \varepsilon_2 = 32.05°}$; $\boldsymbol{P_1, P_2 = 178.46}$ Н; $\boldsymbol{P_3, P_4 = 178.46}$ 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 БВС находится в вертолётном режиме, необходимо написать программы: систему стабилизации и систему траекторного управления для продольного канала движения БВС в указанные координаты (высота, дальность).

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.

По умолчанию БВС сбалансирован на начальной высоте.

В рамках данной задачи для осуществления стабилизации и траекторного управления БВС необходимо будет построить соответствующие ПИД-регуляторы.

Система стабилизации:

- угол тангажа $\vartheta - M_z^u$;
- высоты $H - P_y^u$.

Система траекторного управления:

- продольная скорость $V_{xg} - V_x^u$.

Структура автоматического управления БВС типа конвертоплан в вертолётном режиме для продольного движения показана рисунке 1.

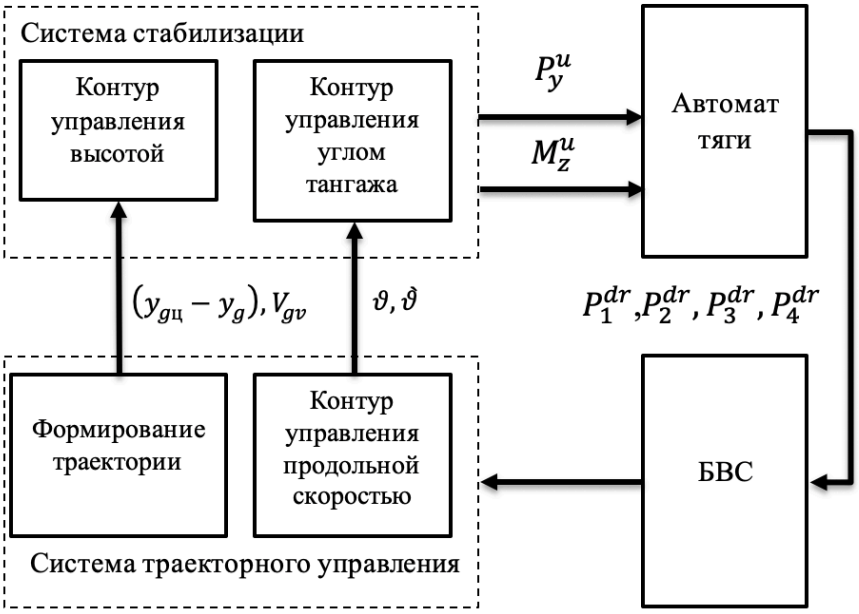


Рисунок 1. Структура САУ продольным движением БВС в вертолётном режиме управления.

Реализовать алгоритм возможно при помощи пропорционально-интегрально-дифференциального регулятора (ПИД-регулятора), структурная схема котрого показана на рисунке 2.

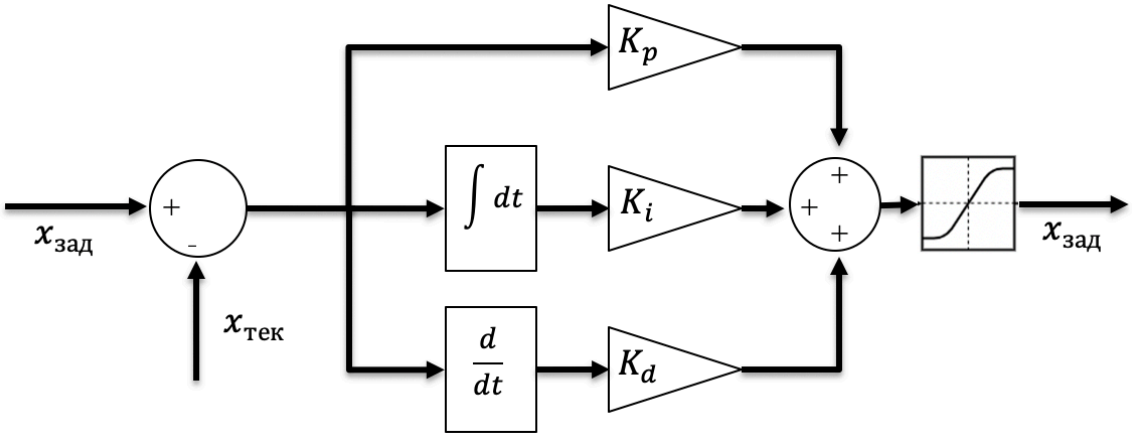


Рисунок 2. Структурная схема ПИД-регулятора.

Как показано на рисунке 2, 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 3:

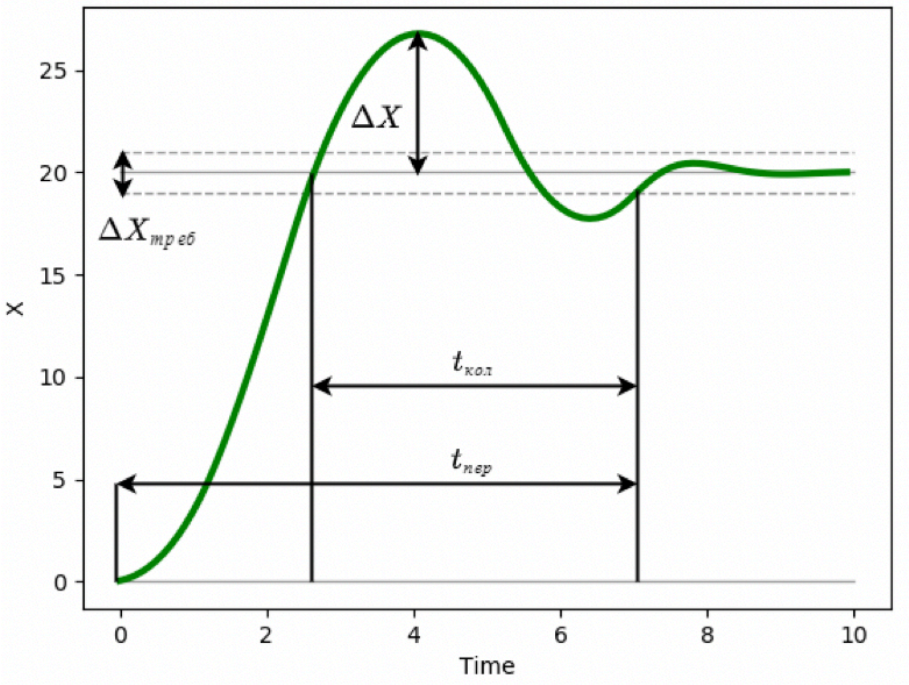


Рисунок 3. Переходный процесс регулируемой величины.

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).

Задачей траекторного управления является перемещение БВС из его текущего местоположения, определяющего отклонение БВС от линии пути А-Б, в точку Б путём сокращения продольного $l_{\text{цп}}$. Так как в задаче рассматривается только продольное движение, угол курса $\psi = \psi_{\text{пот}} = const = 0$, соответственно дальность до точки Б $L_{\text{ц}} = l_{\text{цп}}$ – сокращение $l_{\text{цп}}$ производится за счёт изменения угла тангажа ϑ и последующего появления продольной компоненты линейной скорости V_{xg} (рисунок 4).

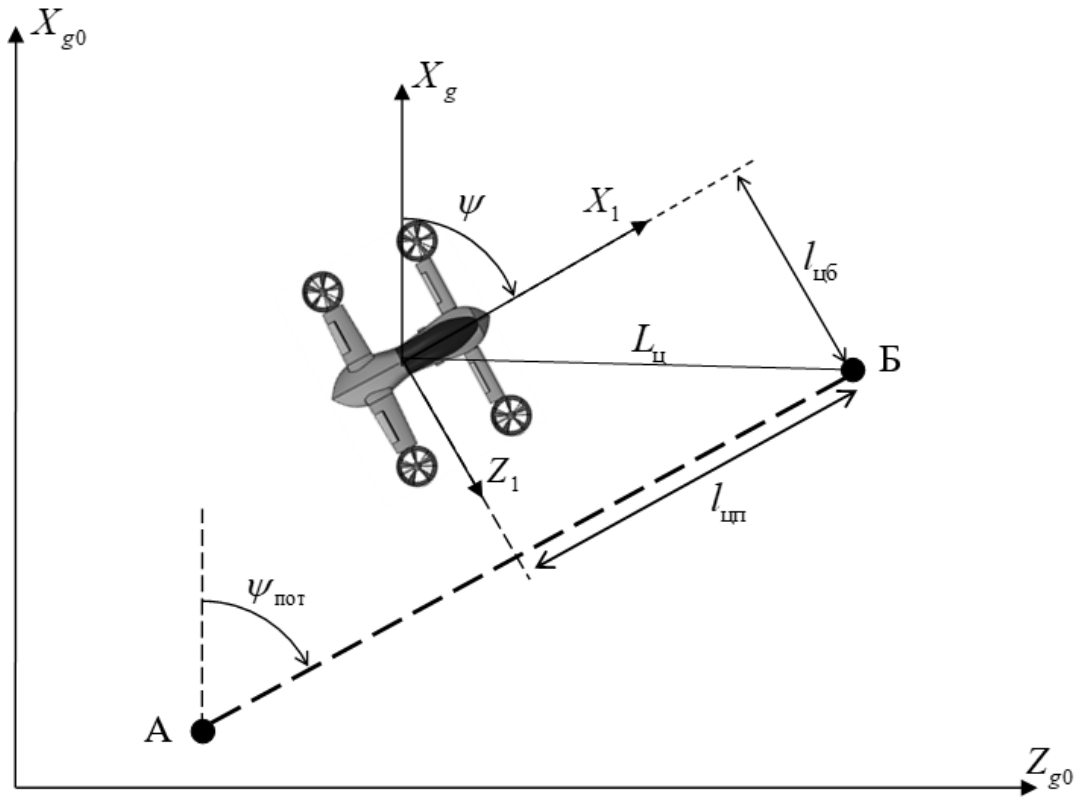


Рисунок 4. Параметры характеризующие расположения БВС относительно линии пути А–Б.

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

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

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
```

Участникам олимпиады рекомендуется производить вычисления в функциях-шаблонах:

- calc_stab – шаблонная функция исполнения системы стабилизации;
- calc_traj – шаблонная функция исполнения системы траекторного управления.

Условия моделирования полёта БВС:

- начальное значение угла тангажа Tang = 0 град.;
- начальное значение угла наклона траектории Theta = 0 град.;
- начальное значение приборной скорости Vpr = 0 км/ч;
- начальное значение высоты БВС H = 50 м.

Заданные параметры полёта БВС:

- заданное дальность до точки L_zad = 50 м;

- заданное высота точки H_zad = 20 м.

Требования к переходному процессу заданного угла тангажа:

	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	$t_{пер} \le 23$	$23 < t_{пер} \le 27$	$t_{пер} > 27$
Перерегулирование	$\Delta X \le 23$	$23 < \Delta X \le 27$	$\Delta X > 27$

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

        self.Vy_prev = 0
        self.VyI = 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 и высоты
    # По умолчанию стабилизирует БВС на заданной высоте.
    def stab_process(self):
        # Заданные значения
        H_zad = self.H_zad
        # Параметры которые могут понадобится для расчета
        # Текущая приборная скорость (можно считать что
горизонтальная), м/с
        Vx = self.u_input.Vw/3.6
        # Текущий угол тангажа, град.
```

```
        Tang = self.u_input.Tan
        # Текущая угловая скорость по оси OZ, в связанной СК,
град.
        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
        # Текущее положение угла поворота ВМГ, рад.
        eps = radians(self.u_output.eps)
        # Плечи
        lz1 = self.lz1
        lz2 = self.lz2

        ly1 = self.ly1
        ly2 = self.ly2

        sinE = sin(eps)
        cosE = cos(eps)
        L_p =   self.L_zad - self.u_input.L
        #
        P_trim_sum =  self.m*9.87/cos(Tang)
        # Требуемая тяга ДВС для поддержания БВС в состоянии
стабилизации на текущей высоте.

        Y_ksu = H_now - H_zad
        Vy_tr = - Y_ksu  / max(3,L_p)
        Vy_zad = clamp(180*Vy_tr,-7,7)
        ay = (Vy - self.Vy_prev)/self.dt
        self.Vy_prev = Vy

        self.VyI =  self.VyI + (Vy - self.Vy_prev) * self.dt

        R = P_trim_sum -  0*self.VyI - 6.5*( Vy - Vy_zad) +
50.5*ay

        k12 = lz2/(2*(lz1 + lz2))
        k34 = lz1/(2*(lz1 + lz2))

        P12_zad = (k12)*(R)
        P34_zad = (k34)*(R)

        dr12 = np.interp(P12_zad,self.P_one,self.drossel)
        dr34 = np.interp(P34_zad,self.P_one,self.drossel)

        self.P_reg_12 =  dr12
        self.P_reg_34 =  dr34


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

```
        self.u_input = input
        # Вызов шаблонной функции
        self.stab_process()
        # 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
```