航天飞行动力学
第三次作业 ——飞行方案设计

一、题目

1. 导弹参数：

* 导弹质量 $m_0 = 320kg$
* 发动机推力 $P = 2000N$
* 初始速度 $V_0 = 250m/s$
* 初始位置 $x_0 = 0m$
* 初始高度 $H_0 = 7000m$
* 初始弹道倾角 $\theta = 0°$
* 初始俯仰角 $\varphi_0 = 0°$
* 初始攻角 $\alpha_0 = 0°$
* 初始俯仰角速度 $\dot{\varphi}_0 = 0rad/\mathrm{s}$
* 初始速度 $V_0 = 250m/s$
* 参考长度 $S_{ref} = 0.45m^2$
* 参考面积 $L_{ref} = 2.5m$
* 升力系数 $C_y = 0.25\alpha + 0.05\delta_z$
* 阻力系数 $C_x = 0.2 + 0.005\alpha^2$
* 俯仰力矩系数 $m_z = -0.1\alpha + 0.024\delta_z$

2. 大气密度计算公式:

$$\begin{cases} \rho_0 = 1.2495\ kg/m^3 \\ T_0 = 288.15K \\ T = T_0 - 0.0065H \\ \rho = \rho_0 \left(\frac{T}{T_0}\right)^{4.25588} \end{cases} \tag{1}$$

3. 飞行方案：

(1) 当 $x < 9100m$ 时，采用瞬时平衡假设

$$\begin{cases} H^* = 2000 \times \cos(0.000314 \times 1.1 \times x) + 5000 \\ \delta_z = k_\varphi \times (H \text{ -} H^*) + k_\varphi \times (H \text{ -} H^*) \\ \delta_z = k_\varphi(H - H^*) + \dot{k}_\varphi H \\ m_s = 0.0kg/s \end{cases} \tag{2}$$

(2) 当 $24000m > x > 9100m$ 时，等高飞行方案，采用瞬时平衡假设。

$$\begin{cases} H^* = 3050m \\ \delta_z = k_\varphi(H - H^*) + \dot{k}_\varphi H \\ \delta_z = k_\varphi(H - H^*) + \dot{k}_\varphi H \\ m_s = 0.46kg/s \end{cases} \tag{3}$$

(3) 当 $x > 24000m\&\&y > 0$，目标位置为 $x_m = 30000m$, 采用比例导引法和瞬时平衡假设

$$\begin{cases} x_m = 30000m \\ m_z^\alpha \alpha + m_z^{\delta_z}\delta_z = 0 \\ m_s = 0.0kg/s \end{cases} \tag{4}$$

注：舵偏角约束 $|\delta_z| \leq 30^\circ$

二、公式推导

1.$x < 24000m$ 的飞行方案：

基于“瞬时平衡”假设，将包含 20 个方程的导弹运动方程组简化为铅垂平面内的质心运动方程组。

$$\begin{cases} m\frac{\mathrm{d}V}{\mathrm{d}t} = P\cos\alpha - X - mg\sin\theta \\ mV\frac{\mathrm{d}\theta}{\mathrm{d}t} = P\sin\alpha + Y - mg\cos\theta \\ \frac{\mathrm{d}x}{\mathrm{d}t} = V\cos\theta \\ \frac{dy}{dt} = V\sin\theta \\ \frac{dm}{dt} = -m_s \\ \alpha_b = -\frac{m_z^{\delta_z}}{m_z^\alpha}\delta_{zb} \\ \delta_z = k_\varphi\left(H - H^*\right) + \dot{k}_\varphi\left(\dot{H} - \dot{H}^*\right) \\ H^* = 2000 \times \cos\left(0.000314 \times 1.1 \times x\right) + 5000 \end{cases} \tag{5}$$

代入各物理量定义式：

$$\begin{cases}\frac{\mathrm{d}V}{\mathrm{d}t}=\frac{P\cos\alpha-X}{m}-g\sin\theta\\ \frac{\mathrm{d}\theta}{\mathrm{d}t}=\frac{P\sin\alpha+Y}{m}-\frac{g\cos\theta}{V}\\ \frac{\mathrm{d}x}{\mathrm{d}t}=V\cos\theta\\ \frac{dy}{dt}=V\sin\theta\\ \frac{dm}{dt}=-m_s\\ \alpha_b=-\frac{m_z^{\delta_z}}{m_z^{\alpha}}\delta_{zb}\\ \delta_z=k_\varphi\left(H-H^*\right)+\dot{k}_\varphi\left(\dot{H}-\dot{H}^*\right)\\ H^*=2000\times\cos\left(0.000314\times1.1\times x\right)+5000\\ Y=\left(0.25\alpha+0.05\delta_z\right)\times\frac{1}{2}\rho V^2\times S_{ref}\\ X=\left(0.2+0.005\alpha^2\right)\times\frac{1}{2}\rho V^2\times S_{ref}\end{cases}\tag{6}$$

$2.x>24000m$ **的飞行方案：**

(1) 末段第一种计算方法：

$$\begin{cases}r\frac{dq}{dt}=V_m\times\sin\eta-V_T\sin\eta_T\\ \tan q=\frac{y_T-y_m}{x_T-x_m}\\ \frac{d\theta^*}{dt}=k\frac{dq}{dt}\\ \theta^*-\theta_0=k(q-q_0)\\ \theta_0,q_0?\\ \delta_z=k_\theta(\theta-\theta^*)+k_{\dot{\theta}}(\dot{\theta}-\dot{\theta}^*)\end{cases}\tag{7}$$

(2) 末段第二种计算方法：

只需要给出比例导引系数根据运动学方程

$$\begin{cases}r\frac{dq}{dt}=V_m\times\sin\eta\ -V_T\sin\eta_T\\ \tan q=\frac{y_T-y_m}{x_T-x_m}\\ \frac{dq}{dt}=\frac{-V_m\sin(\theta-q)}{r}\end{cases}\tag{8}$$

由比例导引法 $\dot{\theta}^*=k\dot{q}$, 可得动力学方程第二式

$$mV_m\dot{\theta}^*=P\sin\alpha+Y-mg\cos\theta\Rightarrow mV_mk\dot{q}=P\sin\alpha+Y-mg\cos\theta\tag{9}$$

由于攻角较小，进行线性化可得

$$mV_mk\dot{q}=P\alpha+Y^\alpha\alpha+Y^{\delta_z}\delta_z-mg\cos\theta\tag{10}$$

由于瞬时平衡 $m_z=0$, 可得

$$-0.1\alpha+0.024\delta_{\bar{z}}=0\Rightarrow\delta_{\bar{z}}=0.1\alpha/0.024\tag{11}$$

代入，可得

$$\alpha=\frac{mV_m k\dot{q}+mg\cos\theta}{P+Y^{\alpha}+Y^{\delta_z}(0.1/0.024)}\Rightarrow\frac{mV_m k\dot{q}+mg\cos\theta}{P+C_y^{\alpha}qS_{ref}+C_y^{\delta_z}qS_{ref}(0.1/0.024)}\tag{12}$$

最后得到弹道方程为

$$\begin{cases}\frac{dV}{dt}=\frac{P\cos\alpha-X}{m}-g\sin\theta\\ \alpha=\frac{mVk\dot{q}+mg\cos\theta}{P+C_y^{\alpha}qS_{ref}+C_y^{\delta_z}qS_{ref}(0.1/0.024)}\\ \frac{dx}{dt}=V\cos\theta\\ \frac{dy}{dt}=V\sin\theta\\ \dot{\theta}^{*}=k\dot{q}\\ \dot{\theta}^{*}=\dot{\theta}\\ \tan q=\frac{y_T-y_m}{x_T-x_m}\\ \frac{dq}{dt}=\frac{-V\sin(\theta-q)}{r}\\ \delta_z=0.1\alpha/0.024\end{cases}\tag{13}$$

补充约束条件

$$\begin{cases}\frac{dV}{dt}=\frac{P\cos\alpha-X}{m}-g\sin\theta\\ \frac{d\theta}{dt}=\frac{-kV\sin(\theta-\arctan\frac{y_T-y_m}{x_T-x_m})}{r}\\ \frac{dx}{dt}=V\cos\theta\\ \frac{dy}{dt}=V\sin\theta\\ \frac{dm}{dt}=-m_s\\ \alpha=\frac{mV\dot{\theta}+mg\cos\theta}{P+C_y^{\alpha}qS_{ref}+C_y^{\delta_z}qS_{ref}(0.1/0.024)}\\ \alpha=-\frac{m_z^{\delta_z}}{m_z^{\alpha}}\delta_z\\ |\delta_z|\le 30^{\circ}\end{cases}\tag{14}$$

三、仿真结果

三个系数的取值：

$K_{\varphi}=-0.5$

$\dot{K}_{\varphi}=0.6*K_phi$

$K_3=5$

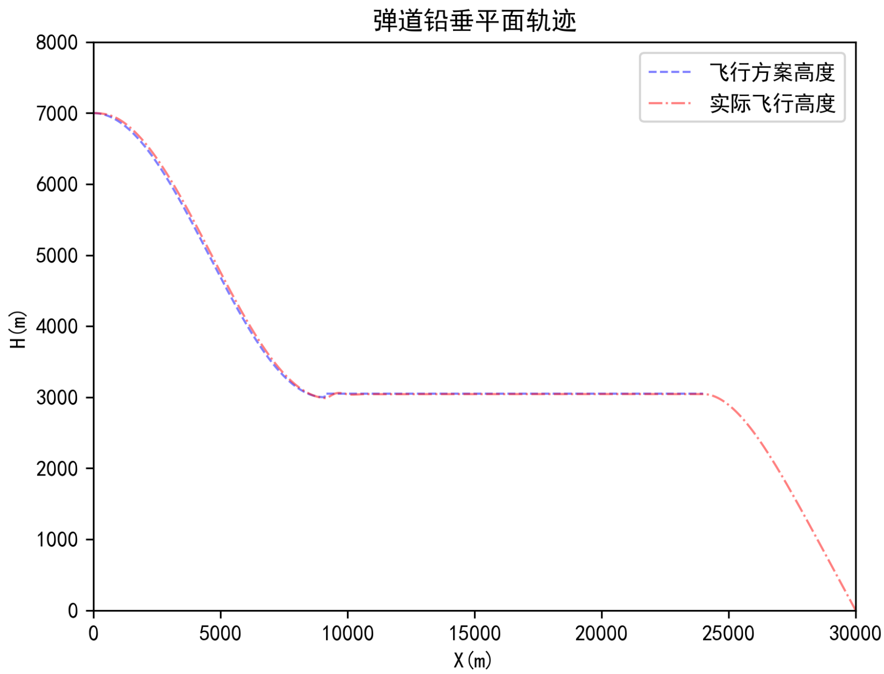

图 1 导弹飞行轨迹图

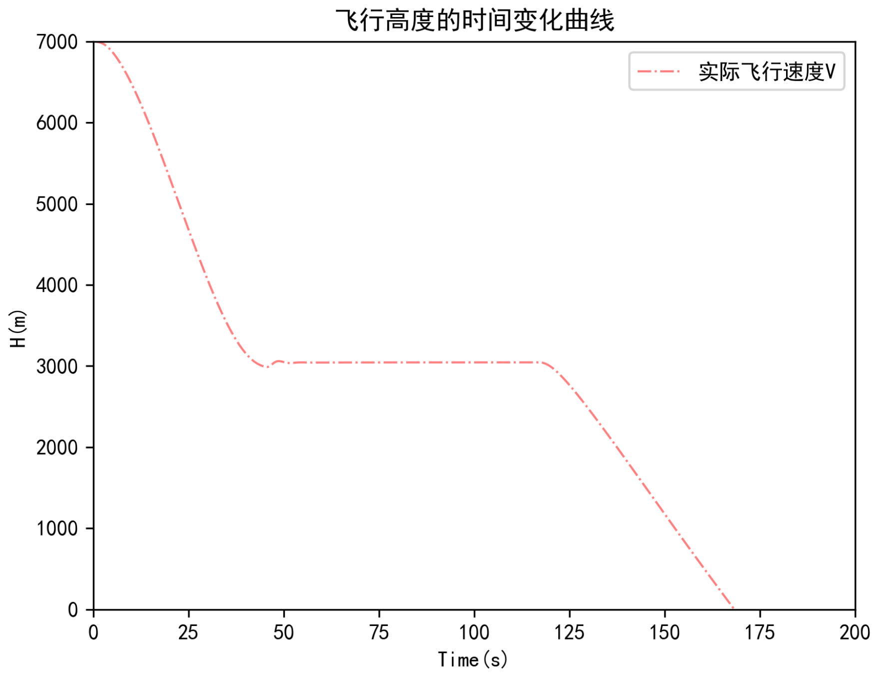

图 2 导弹飞行高度的时间曲线图

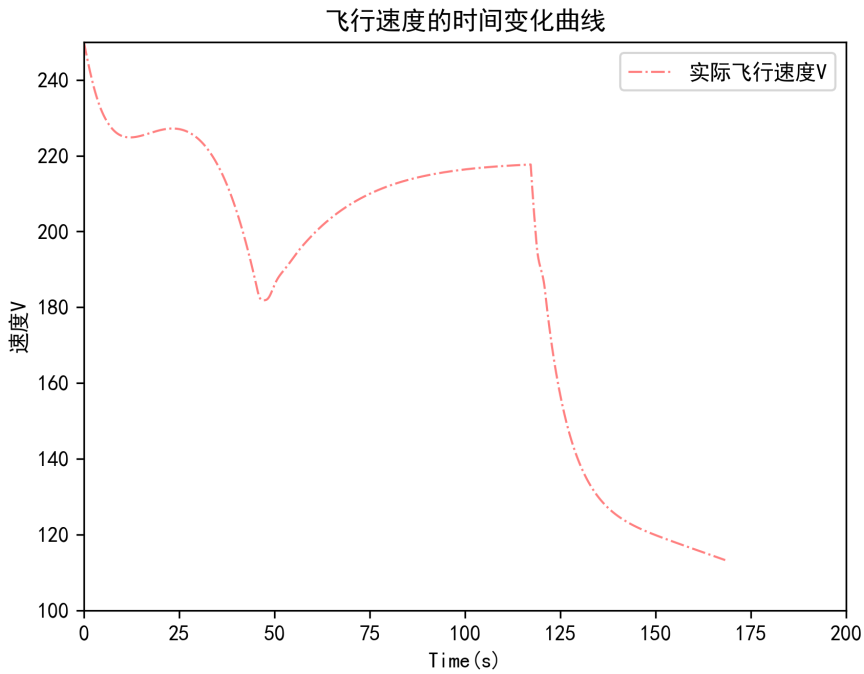

图 3 导弹飞行速度的时间曲线图

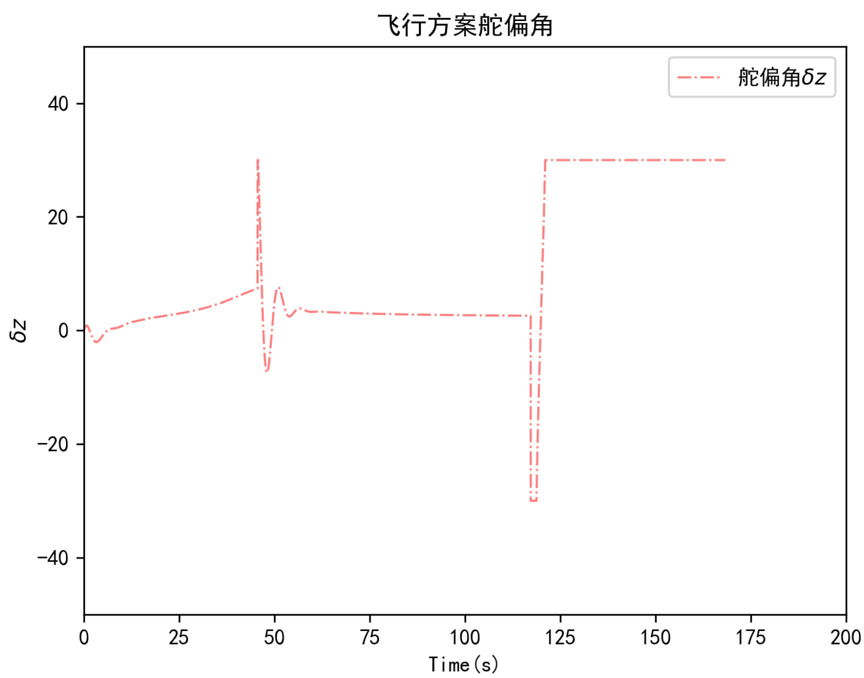

图 4 导弹飞行舵偏角的时间曲线图

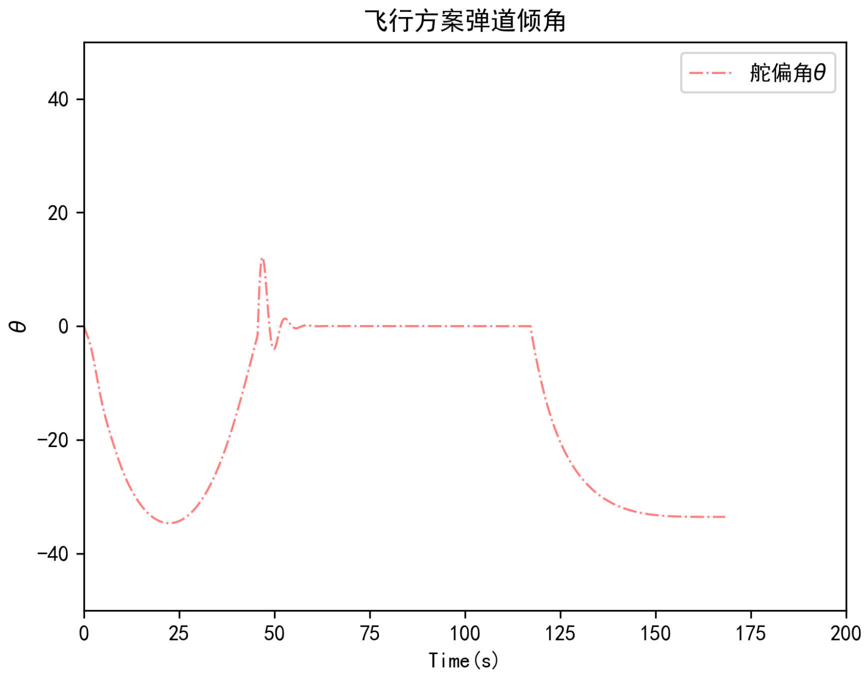

图 5 导弹飞行弹道倾角的时间曲线图

四、结果分析

1.k_φ 的影响：

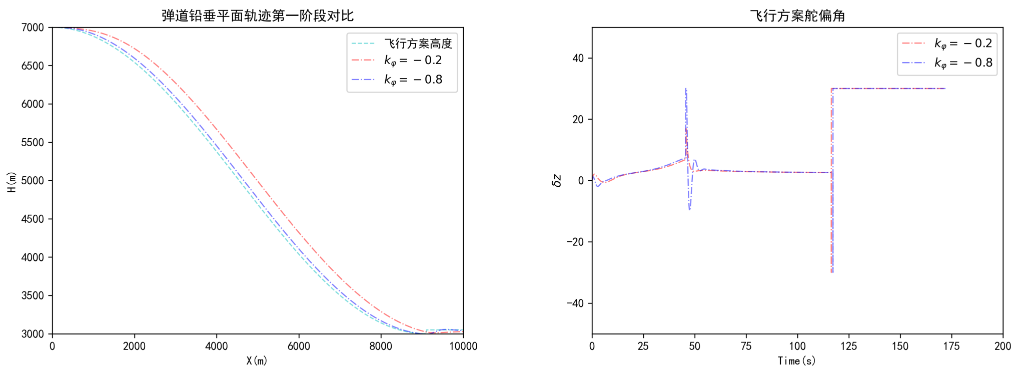

图 6 K_φ 大小对导弹飞行弹道和舵偏角的影响

如图 6所示，K_φ 是理想控制方程的放大系数，K_φ 绝对值越大，导弹越快地恢复到预定的飞行方案。

2.$\dot{K}_\varphi$ 的影响：

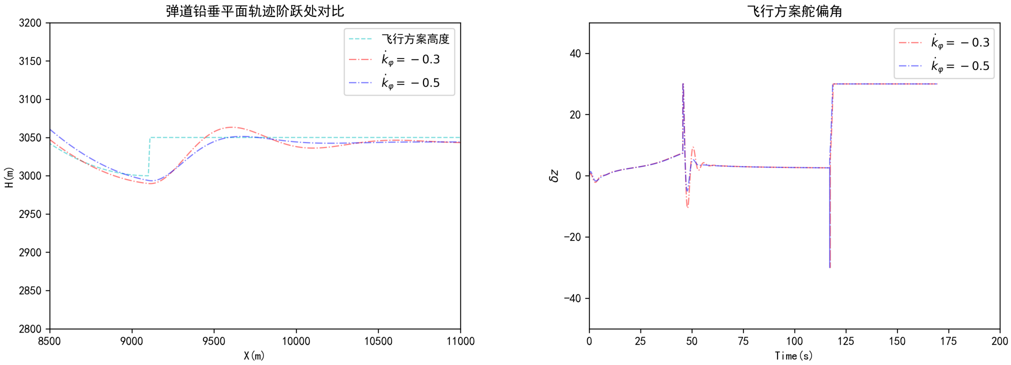

图 7 $\dot{K}_\varphi$ 大小对导弹飞行弹道和舵偏角的影响

如图 7所示，$\dot{K}_\varphi$ 是用于减小超调量的放大系数，起到阻尼作用，k_φ 绝对值越大，导弹越快地稳定到预定的飞行方案。

3.k_3 的影响：

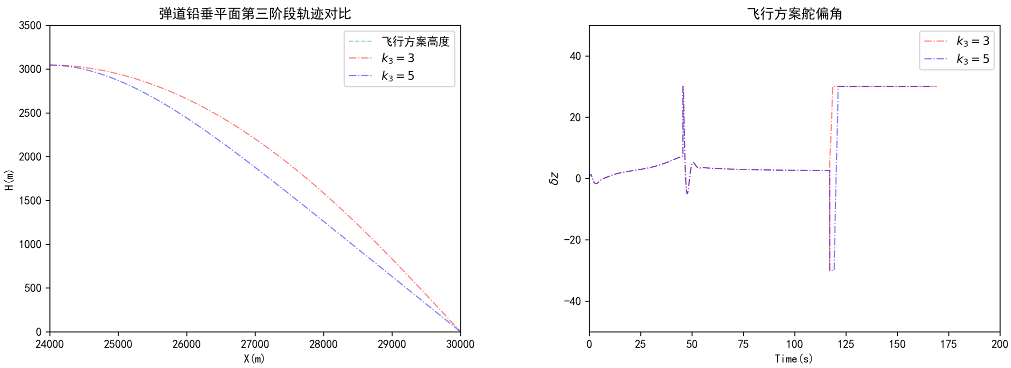

图 8 k_3 大小对导弹飞行弹道和舵偏角的影响

如图 8所示，K_3 是比例导引法的比例系数，K_3 越大，后期弹道约平直。

注：第三阶段的舵偏角由于使用显式的 Euler 法计算，结果有一定的发散。

源代码 1: **main.py**

```python
""""""
弹道计算程序
"""

import numpy as np
from matplotlib import pyplot as plt

# 展示高清图
from matplotlib_inline import backend_inline
backend_inline.set_matplotlib_formats('svg')

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

# 导弹参数
S_ref    =   0.45
L_ref    =   2.5

# 放大系数
K_phi = - 0.5
K_phi_dot= 0.6* K_phi
K_q = 5


# 仿真时间步
timestep = 0.01

# 导弹状态定义
class statu():
    __slot__=['Time','X','H','V','theta','mass','alpha','deltaz']

    # 位置
    # 速度
    # 欧拉角
    # 角加速度
    # 舵偏角

    # 初始化
    def __init__(self, Time, X=0, H=0, V=0, theta=0, mass=0):
        self.Time = Time
        self.X = X
```

```
        self.H = H
        self.V = V
        self.theta = theta
        self.mass = mass
        self.alpha = 0
        self.deltaz = 0
        self.dq = 0

    # 显式Euler法，给定飞行高度
    def Euler(self, before, dmass):
        self.Time = before.Time + timestep

        self.X = before.X + before.V * np.cos(before.theta) * timestep
        self.H = before.H + before.V * np.sin(before.theta) * timestep

        self.deltaz = K_phi * (self.H - High_goal(self.X)) + K_phi_dot* (before.V * np
            .sin(before.theta) - High_goal_dot(self.X))

        if self.deltaz > 30:
            self.deltaz = 30
        elif self.deltaz < -30:
            self.deltaz = -30

        self.alpha =  0.24 *self.deltaz

        Y = (0.25 * self.alpha + 0.05* self.deltaz) * 0.5 * air(self.H) * before.V *
            before.V * S_ref

        X = (0.005 * self.alpha * self.alpha + 0.2) * 0.5 * air(self.H) * before.V *
            before.V * S_ref

        self.mass = before.mass - dmass * timestep
        if dmass == 0:
            P = 0
        else:
            P = 2000

        self.V = before.V + (P*np.cos(before.alpha*3.14159625/180) - X - self.mass
            *9.8*np.sin(before.theta)) /self.mass*timestep
        self.theta = before.theta + (P*np.sin(self.alpha*3.14159625/180) + Y - self.
            mass*9.8*np.cos(before.theta)) /self.mass/self.V*timestep

    # 比例导引法，给定目标位置
```

```
    def Euler2(self, before, Xm, Ym):
        self.Time = before.Time + timestep

        self.X = before.X + before.V * np.cos(before.theta) * timestep
        self.H = before.H + before.V * np.sin(before.theta) * timestep
        self.mass = before.mass

        self.r = np.sqrt((self.X - Xm)*(self.X - Xm) + (self.H - Ym)*(self.H - Ym))

        self.dq = - before.V * np.sin(before.theta - np.arctan(( self.H - Ym)/(self.X
            - Xm)))/ self.r

        self.theta = before.theta + K_q * self.dq * timestep

        P = 0


        self.alpha = (self.mass* before.V * K_q * self.dq + self.mass * 9.8 * np.cos(
            self.theta))/(P +  (0.25 + 0.05/0.24) * 0.5 * air(self.H) * before.V *
            before.V * S_ref) /3.14159*180

        self.deltaz = self.alpha / 0.24

        if self.deltaz > 30:
            self.deltaz = 30
        if self.deltaz < -30:
            self.deltaz = -30

        self.alpha =  0.24 *self.deltaz

        X = (0.005 * self.alpha * self.alpha + 0.2) * 0.5 * air(self.H) * before.V *
            before.V * S_ref
        self.V = before.V + (P*np.cos(before.alpha*3.14159625/180) - X - self.mass
            *9.8*np.sin(before.theta)) /self.mass*timestep

    def Euler3(self, before, Xm, Ym):
        self.Time = before.Time + timestep

        self.X = before.X + before.V * np.cos(before.theta) * timestep
        self.H = before.H + before.V * np.sin(before.theta) * timestep
        self.mass = before.mass

        self.r = np.sqrt((self.X - Xm)*(self.X - Xm) + (self.H - Ym)*(self.H - Ym))
```

```python

        self.dtheta = - K_q * before.V * np.sin(before.theta - np.arctan(( self.H - Ym)/(self.X - Xm)))/ self.r

        self.theta = before.theta + self.dtheta * timestep

        P = 0

        self.alpha = (self.mass* before.V * self.dtheta + self.mass * 9.8 * np.cos(self.theta))/(P +  (0.25 + 0.05/0.24) * 0.5 * air(self.H) * before.V * before.V * S_ref) /3.14159*180

        self.deltaz = self.alpha / 0.24

        if self.deltaz > 30:
            self.deltaz = 30
        if self.deltaz < -30:
            self.deltaz = -30

        self.alpha =  0.24 * self.deltaz

        X = (0.005 * self.alpha * self.alpha + 0.2) * 0.5 * air(self.H) * before.V * before.V * S_ref
        self.V = before.V + ((P*np.cos(self.alpha*3.14159/180) - X)/self.mass-9.8*np.sin(self.theta)) *timestep

# 大气参数
def air (High):
    rho0 =1.2495
    T0  = 288.15
    Temp = T0 - 0.0065*High
    rho = rho0 * np.exp(4.25588*np.log(Temp / T0))
    return rho

# 飞行方案
def High_goal(X):
    if X <= 9100:
        return 2000 * np.cos(0.000314 * 1.1 * X) + 5000
    elif X <= 24000:
        return 3050
    else:
        return 0

```

```python
# 飞行方案的时间导数
def High_goal_dot(X):
    if X <= 9100:
        return -2000 * 0.000314 * np.sin(0.000314 * 1.1 * X)
    elif X <= 24000:
        return 0
    else:
        return 0


# 飞行初始状态
statu_n = [statu(0, 0, 7000, 250, 0, 320)]
statu_n[0].alpha = 0
statu_n[0].deltaz = 0

X_goal = np.arange(0,24000,10)
H_goal = [High_goal(i) for i in X_goal]
plt.plot(X_goal,H_goal, 'b--', alpha=0.5, linewidth=1, label='飞行方案高度')

# 第一阶段
while statu_n[-1].X < 9100:
    statu_n.append(statu(statu_n[-1].Time + timestep))
    statu_n[-1].Euler(statu_n[-2],0)
    #print(statu_n[-1].alpha)

# 第二阶段
while statu_n[-1].X <= 24000:
    statu_n.append(statu(statu_n[-1].Time + timestep))
    statu_n[-1].Euler(statu_n[-2],0.46)
    #print(statu_n[-1].theta)

# 第三阶段
while statu_n[-1].X <= 30000 and statu_n[-1].H > 0:
    statu_n.append(statu(statu_n[-1].Time + timestep))
    statu_n[-1].Euler2(statu_n[-2],30000,0)
    #print(statu_n[-1].V)


# 绘图
X_data = [n.X for n in statu_n]
H_data = [n.H for n in statu_n]
plt.plot(X_data,H_data, 'r-.', alpha=0.5, linewidth=1, label='实际飞行高度')
plt.title("弹道铅垂平面轨迹")
```

```
plt.legend()  #显示上面的label
plt.xlabel('X(m)') #x_label
plt.ylabel('H(m)')#y_label
plt.ylim(0,8000)
plt.xlim(0,30000) #仅设置y轴坐标范围
plt.savefig('img/飞行轨迹.png', dpi=300)
plt.clf()

T_data = [n.Time for n in statu_n]
deltaz_data = [n.deltaz for n in statu_n]
plt.plot(T_data,deltaz_data, 'r-.', alpha=0.5, linewidth=1, label='舵偏角$\delta z$')
plt.title("飞行方案舵偏角")
plt.legend()  #显示上面的label
plt.xlabel('Time(s)') #x_label
plt.ylabel('$\delta z$')#y_label
plt.ylim(-50,50)
plt.xlim(0,200)
plt.savefig('img/飞行舵偏角.png', dpi=300)
plt.clf()

plt.plot(T_data,H_data, 'r-.', alpha=0.5, linewidth=1, label='实际飞行速度V')
plt.title("飞行高度的时间变化曲线")
plt.legend()  #显示上面的label
plt.xlabel('Time(s)') #x_label
plt.ylabel('H(m)')#y_label
plt.ylim(0,7000)
plt.xlim(0,200)
plt.savefig('img/飞行高度.png', dpi=300)
plt.clf()

V_data = [n.V for n in statu_n]
plt.plot(T_data,V_data, 'r-.', alpha=0.5, linewidth=1, label='实际飞行速度V')
plt.title("飞行速度的时间变化曲线")
plt.legend()  #显示上面的label
plt.xlabel('Time(s)') #x_label
plt.ylabel('速度V')#y_label
plt.ylim(100,250)
plt.xlim(0,200)
plt.savefig('img/飞行速度.png', dpi=300)
plt.clf()

theta_data = [n.theta*180/3.14159 for n in statu_n]
plt.plot(T_data,theta_data, 'r-.', alpha=0.5, linewidth=1, label=r'舵偏角$\theta$')
```

```
plt.title("飞行方案弹道倾角")
plt.legend()  #显示上面的label
plt.xlabel('Time(s)') #x_label
plt.ylabel(r'$\theta$')#y_label
plt.ylim(-50,50)
plt.xlim(0,200)
plt.savefig('img/飞行弹道倾角.png', dpi=300)
plt.clf()
```

源代码 2: **comparis.py**

```
"""
弹道计算程序
分析放大系数的影响
"""

import numpy as np
from matplotlib import pyplot as plt

# 展示高清图
from matplotlib_inline import backend_inline
backend_inline.set_matplotlib_formats('svg')

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

# 导弹参数
S_ref   =   0.45
L_ref   =   2.5

# 仿真时间步
timestep = 0.01

# 导弹状态定义
class statu():
    __slot__=['Time','X','H','V','theta','mass','alpha','deltaz']

    # 位置
    # 速度
    # 欧拉角
    # 角加速度
    # 舵偏角

    # 初始化
    def __init__(self, Time, X=0, H=0, V=0, theta=0, mass=0):
        self.Time = Time
        self.X = X
        self.H = H
        self.V = V
        self.theta = theta
        self.mass = mass
        self.alpha = 0
```

```python
        self.deltaz = 0
        self.q = 0

    # 显式Euler法，给定飞行高度
    def Euler(self, before, dmass,K_phi, K_phi_dot):
        self.Time = before.Time + timestep

        self.X = before.X + before.V * np.cos(before.theta) * timestep
        self.H = before.H + before.V * np.sin(before.theta) * timestep

        self.deltaz = K_phi * (self.H - High_goal(self.X)) + K_phi_dot* (before.V * np.sin(before.theta) - High_goal_dot(self.X))

        if self.deltaz > 30:
            self.deltaz = 30
        elif self.deltaz < -30:
            self.deltaz = -30

        self.alpha =  0.24 *self.deltaz

        Y = (0.25 * self.alpha + 0.05* self.deltaz) * 0.5 * air(self.H) * before.V * before.V * S_ref

        X = (0.005 * self.alpha * self.alpha + 0.2) * 0.5 * air(self.H) * before.V * before.V * S_ref

        self.mass = before.mass - dmass * timestep
        if dmass == 0:
            P = 0
        else:
            P = 2000

        self.V = before.V + (P*np.cos(before.alpha*3.14159625/180) - X - self.mass*9.8*np.sin(before.theta)) /self.mass*timestep
        self.theta = before.theta + (P*np.sin(self.alpha*3.14159625/180) + Y - self.mass*9.8*np.cos(before.theta)) /self.mass/self.V*timestep

    # 比例导引法，给定目标位置
    def Euler2(self, before, Xm, Ym, K_q):
        self.Time = before.Time + timestep

        self.X = before.X + before.V * np.cos(before.theta) * timestep
        self.H = before.H + before.V * np.sin(before.theta) * timestep
```

```
        self.mass = before.mass

        self.r = np.sqrt((self.X - Xm)*(self.X - Xm) + (self.H - Ym)*(self.H - Ym))

        self.dq = - before.V * np.sin(before.theta - np.arctan(( self.H - Ym)/(self.X
            - Xm)))/ self.r

        self.theta = before.theta + K_q * self.dq * timestep

        P = 0


        self.alpha = (self.mass* before.V * K_q * self.dq + self.mass * 9.8 * np.cos(
            self.theta))/(P +  (0.25 + 0.05/0.24) * 0.5 * air(self.H) * before.V *
            before.V * S_ref) /3.14159*180

        self.deltaz = self.alpha / 0.24

        if self.deltaz > 30:
            self.deltaz = 30
        if self.deltaz < -30:
            self.deltaz = -30

        self.alpha =  0.24 *self.deltaz

        X = (0.005 * self.alpha * self.alpha + 0.2) * 0.5 * air(self.H) * before.V *
            before.V * S_ref
        self.V = before.V + (P*np.cos(before.alpha*3.14159625/180) - X - self.mass
            *9.8*np.sin(before.theta)) /self.mass*timestep

# 大气参数
def air (High):
    rho0 =1.2495
    T0  = 288.15
    Temp = T0 - 0.0065*High
    rho = rho0 * np.exp(4.25588*np.log(Temp / T0))
    return rho

# 飞行方案
def High_goal(X):
    if X <= 9100:
        return 2000 * np.cos(0.000314 * 1.1 * X) + 5000
    elif X <= 24000:
```

```python
        return 3050
    else:
        return 0

# 飞行方案的时间导数
def High_goal_dot(X):
    if X <= 9100:
        return -2000 * 0.000314 * np.sin(0.000314 * 1.1 * X)
    elif X <= 24000:
        return 0
    else:
        return 0

def calculate(K_phi ,K_phi_dot, K_q):

    # 飞行初始状态
    statu_n = [statu(0, 0, 7000, 250, 0, 320)]
    statu_n[0].alpha = 0
    statu_n[0].deltaz = 0

    # 第一阶段
    while statu_n[-1].X < 9100:
        statu_n.append(statu(statu_n[-1].Time + timestep))
        statu_n[-1].Euler(statu_n[-2],0,K_phi ,K_phi_dot )

    # 第二阶段
    while statu_n[-1].X <= 24000:
        statu_n.append(statu(statu_n[-1].Time + timestep))
        statu_n[-1].Euler(statu_n[-2],0.46,K_phi ,K_phi_dot)


    # 第三阶段
    while statu_n[-1].X <= 30000 and statu_n[-1].H > 0:
        statu_n.append(statu(statu_n[-1].Time + timestep))
        statu_n[-1].Euler2(statu_n[-2],30000,0,K_q)

    return statu_n

# 飞行方案
X_goal = np.arange(0,24000,10)
H_goal = [High_goal(i) for i in X_goal]

# 放大系数
```

```
statu_1=calculate(-0.2,-0.5,2)
statu_2=calculate(-0.8,-0.5,2)

# 绘图

## 第一个放大系数
X_data_1 = [n.X for n in statu_1]
H_data_1 = [n.H for n in statu_1]
X_data_2 = [n.X for n in statu_2]
H_data_2 = [n.H for n in statu_2]

plt.plot(X_goal,H_goal, 'c--', alpha=0.5, linewidth=1, label='飞行方案高度')

plt.plot(X_data_1,H_data_1, 'r-.', alpha=0.5, linewidth=1, label=r'$k_\varphi=-0.2$')
plt.plot(X_data_2,H_data_2, 'b-.', alpha=0.5, linewidth=1, label=r'$k_\varphi=-0.8$')

plt.title("弹道铅垂平面轨迹第一阶段对比")
plt.legend()  #显示上面的label
plt.xlabel('X(m)') #x_label
plt.ylabel('H(m)')#y_label
plt.ylim(3000,7000)
plt.xlim(0,10000) #仅设置y轴坐标范围
plt.savefig('img/飞行轨迹2.png', dpi=300)
plt.clf()


T_data_1 = [n.Time for n in statu_1]
T_data_2 = [n.Time for n in statu_2]
deltaz_data_1 = [n.deltaz for n in statu_1]
deltaz_data_2 = [n.deltaz for n in statu_2]

plt.plot(T_data_1,deltaz_data_1, 'r-.', alpha=0.5, linewidth=1, label=r'$k_\varphi
    =-0.2$')
plt.plot(T_data_2,deltaz_data_2, 'b-.', alpha=0.5, linewidth=1, label=r'$k_\varphi
    =-0.8$')

plt.title("飞行方案舵偏角")
plt.legend()  #显示上面的label
plt.xlabel('Time(s)') #x_label
plt.ylabel('$\delta z$')#y_label
plt.ylim(-50,50)
plt.xlim(0,200)
plt.savefig('img/飞行舵偏角2.png', dpi=300)
```

```python
plt.clf()

## 第二个放大系数
statu_3 = calculate(-0.6,-0.3,3)
statu_4 = calculate(-0.6,-0.5,3)

X_data_3 = [n.X for n in statu_3]
H_data_3 = [n.H for n in statu_3]
X_data_4 = [n.X for n in statu_4]
H_data_4 = [n.H for n in statu_4]

plt.plot(X_goal,H_goal, 'c--', alpha=0.5, linewidth=1, label='飞行方案高度')

plt.plot(X_data_3,H_data_3, 'r-.', alpha=0.5, linewidth=1, label=r'$\dot{k}_\varphi
    =-0.3$')
plt.plot(X_data_4,H_data_4, 'b-.', alpha=0.5, linewidth=1, label=r'$\dot{k}_\varphi
    =-0.5$')

plt.title("弹道铅垂平面轨迹阶跃处对比")
plt.legend()  #显示上面的label
plt.xlabel('X(m)') #x_label
plt.ylabel('H(m)')#y_label
plt.ylim(2800,3200)
plt.xlim(8500,11000) #仅设置y轴坐标范围
plt.savefig('img/飞行轨迹3.png', dpi=300)
plt.clf()

T_data_3 = [n.Time for n in statu_3]
T_data_4 = [n.Time for n in statu_4]
deltaz_data_3 = [n.deltaz for n in statu_3]
deltaz_data_4 = [n.deltaz for n in statu_4]

plt.plot(T_data_3,deltaz_data_3, 'r-.', alpha=0.5, linewidth=1, label=r'$\dot{k}_\
    varphi=-0.3$')
plt.plot(T_data_4,deltaz_data_4, 'b-.', alpha=0.5, linewidth=1, label=r'$\dot{k}_\
    varphi=-0.5$')

plt.title("飞行方案舵偏角")
plt.legend()  #显示上面的label
plt.xlabel('Time(s)') #x_label
plt.ylabel('$\delta z$')#y_label
plt.ylim(-50,50)
plt.xlim(0,200)
```

```python
plt.savefig('img/飞行舵偏角3.png', dpi=300)
plt.clf()


## 第三个放大系数
statu_5 = calculate(-0.6, -0.5, 3)
statu_6 = calculate(-0.6, -0.5, 6)

X_data_5 = [n.X for n in statu_5]
H_data_5 = [n.H for n in statu_5]
X_data_6 = [n.X for n in statu_6]
H_data_6 = [n.H for n in statu_6]

plt.plot(X_goal,H_goal, 'c--', alpha=0.5, linewidth=1, label='飞行方案高度')

plt.plot(X_data_5,H_data_5, 'r-.', alpha=0.5, linewidth=1, label=r'$k_3=3$')
plt.plot(X_data_6,H_data_6, 'b-.', alpha=0.5, linewidth=1, label=r'$k_3=5$')


plt.title("弹道铅垂平面第三阶段轨迹对比")
plt.legend()  #显示上面的label
plt.xlabel('X(m)') #x_label
plt.ylabel('H(m)')#y_label
plt.ylim(0,3500)
plt.xlim(24000,30000) #仅设置y轴坐标范围
plt.savefig('img/飞行轨迹4.png', dpi=300)
plt.clf()

T_data_5 = [n.Time for n in statu_5]
T_data_6 = [n.Time for n in statu_6]
deltaz_data_5 = [n.deltaz for n in statu_5]
deltaz_data_6 = [n.deltaz for n in statu_6]

plt.plot(T_data_5,deltaz_data_5, 'r-.', alpha=0.5, linewidth=1, label=r'$k_3=3$')
plt.plot(T_data_6,deltaz_data_6, 'b-.', alpha=0.5, linewidth=1, label=r'$k_3=5$')
plt.title("飞行方案舵偏角")
plt.legend()  #显示上面的label
plt.xlabel('Time(s)') #x_label
plt.ylabel('$\delta z$')#y_label
plt.ylim(-50,50)
plt.xlim(0,200)
plt.savefig('img/飞行舵偏角4.png', dpi=300)
plt.clf()
```