**航天飞行器设计大作业**

**——导弹燃料相对质量因数计算**

**一、问题描述**

1、根据已知条件，采用数值积分法求解相对量运动微分方程组，计算燃料相对质量因数。

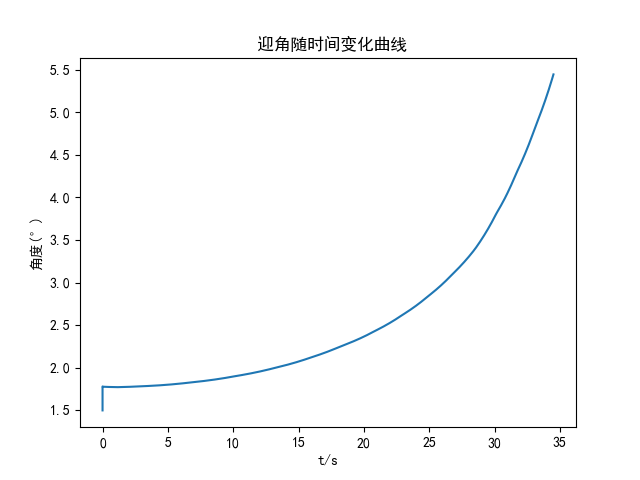
2、综合运用积分、插值计算等计算方法，采用C、C++，或者Matlab等语言的一种，编制计算程序。

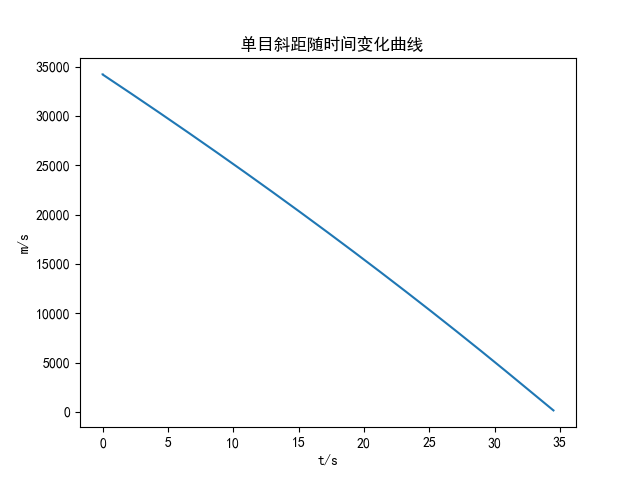
**二、模型建立**

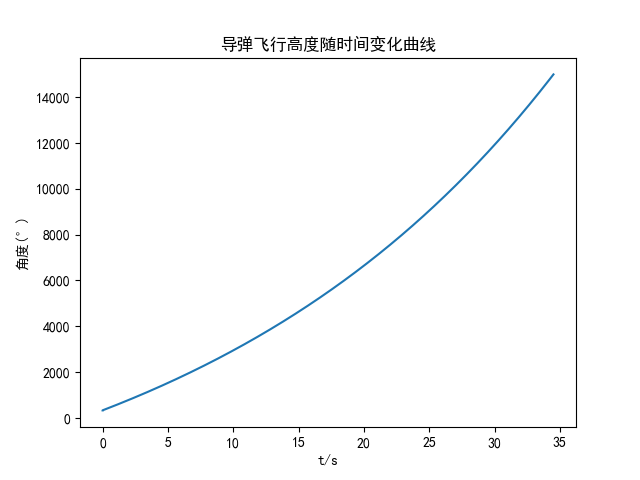


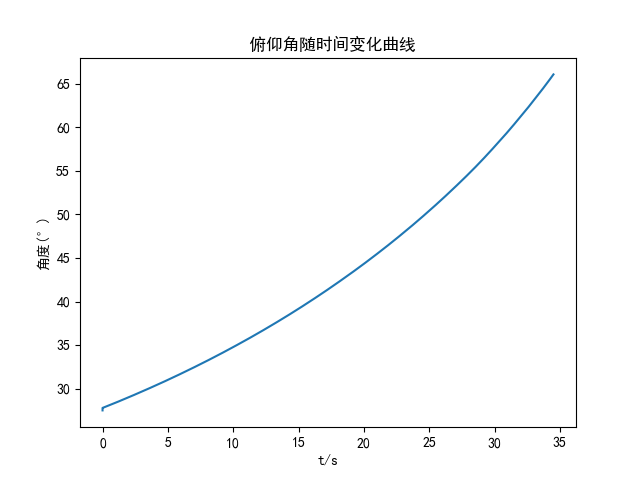
**三、弹道计算**

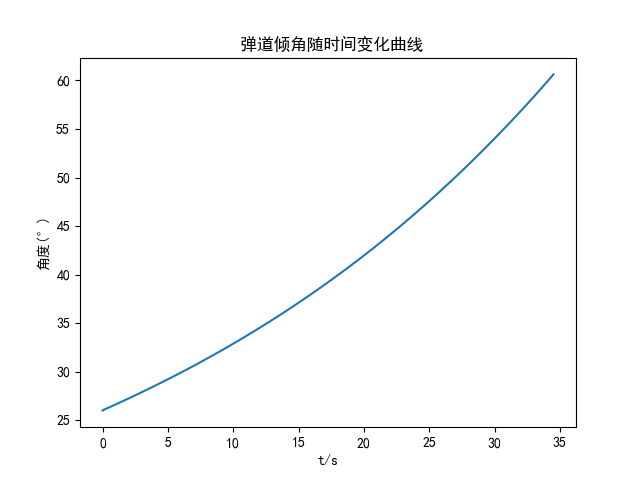
利用python程序可以迭代得到如下的参数变化图像：

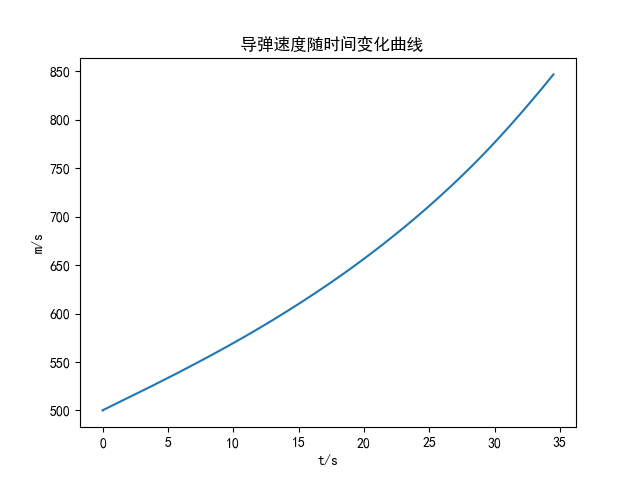


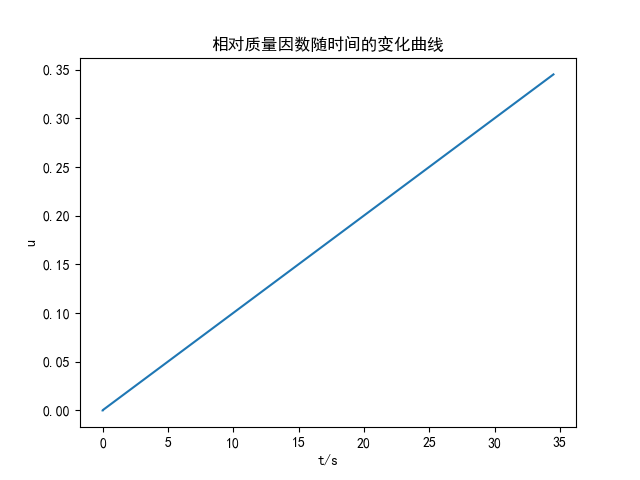












最终计算得导弹的相对质量因数为: 0.3453

**四、python程序如下**

1. rnok.py文件：

import Initialize

import Interpolate

import ShowResult

import numpy

from scipy import interpolate

from numpy import sin,cos,tan

#初始化获得各参数及插值表及迭代初值

(alp1,ma1,cx0,alpha2,ma2,cy0,yh,ph,ah)=Interpolate.interpolation();

(u,t,xt,v,x,y,Drx,cx,cya,alpj,alph,thtj,thth,v0,t0,x0,y0,alp0,tht0,Is,g,P,p0,vt,dt0,yt0,xt0,a,cotq0)=Initialize.initialize(yh,ah);

#迭代参量及迭代步长

i=0;

h=0.0001;

# 龙格库塔迭代

while (y[i]<yt0):

ma=v[i]/a;

cotq=cotq0-u[i]\*Is\*vt/[P\*g\*yt0];

p=interpolate.interp1d(yh,ph)(y[i]);#插值得到密度

a=interpolate.interp1d(yh,ah)(y[i]);#插值得到音速

CCX=interpolate.interp2d(alp1,ma1,cx0,kind='linear');

cx[i] =CCX(alpj[i],ma);

CCYA=interpolate.interp2d(alpha2,ma2,cy0,kind='linear');

cya[i] =180/numpy.pi\*CCYA(alpj[i],ma);

Kv1 = Is / (1 - u[i]) - p \* v[i] \*\* 2 \* cx[i] \* Is / (2 \* P \* p0 \* (1 - u[i])) - Is / P \* sin(thth[i]);

Kv2 = Is / (1 - (u[i] + h / 2)) - p \* (v[i] + h / 2 \* Kv1) \*\* 2 \* cx[i] \* Is / (

2 \* P \* p0 \* (1 - (u[i] + h / 2))) - Is / P \* sin(thth[i]);

Kv3 = Is / (1 - (u[i] + h / 2)) - p \* (v[i] + h / 2 \* Kv2) \*\* 2 \* cx[i] \* Is / (

2 \* P \* p0 \* (1 - (u[i] + h / 2))) - Is / P \* sin(thth[i]);

Kv4 = Is / (1 - (u[i] + h)) - p \* (v[i] + h \* Kv3) \*\* 2 \* cx[i] \* Is / (

2 \* P \* p0 \* (1 - (u[i] + h))) - Is / P \* sin(thth[i]);

Kth1 = vt / yt0 \* (Is / (P \* g) + (

v[i] / (P \* g) \* Is \* v[i] \* sin(thth[i]) - y[i] \* (Kv1 + 2 \* Kv2 + 2 \* Kv3 + Kv4) / 6) / (

v[i] \*\* 2 \* sin(thth[i]))) / (

1 + (cotq0 - u[i] \* Is \* vt / (P \* g \* yt0)) / tan(thth[i]));

Kth2 = vt / yt0 \* (Is / (P \* g) + (v[i] / (P \* g) \* Is \* v[i] \* sin(thth[i] + Kth1 \* h / 2) - y[i] \* (

Kv1 + 2 \* Kv2 + 2 \* Kv3 + Kv4) / 6) / (v[i] \*\* 2 \* sin(thth[i] + Kth1 \* h / 2))) / (

1 + (cotq0 - (u[i] + h / 2) \* Is \* vt / (P \* g \* yt0)) / tan(thth[i] + Kth1 \* h / 2));

Kth3 = vt / yt0 \* (Is / (P \* g) + (v[i] / (P \* g) \* Is \* v[i] \* sin(thth[i] + Kth2 \* h / 2) - y[i] \* (

Kv1 + 2 \* Kv2 + 2 \* Kv3 + Kv4) / 6) / (v[i] \*\* 2 \* sin(thth[i] + Kth2 \* h / 2))) / (

1 + (cotq0 - (u[i] + h / 2) \* Is \* vt / (P \* g \* yt0)) / tan(thth[i] + Kth2 \* h / 2));

Kth4 = vt / yt0 \* (Is / (P \* g) + (

v[i] / (P \* g) \* Is \* v[i] \* sin(thth[i] + Kth3 \* h) - y[i] \* (Kv1 + 2 \* Kv2 + 2 \* Kv3 + Kv4) / 6) / (

v[i] \*\* 2 \* sin(thth[i] + Kth3 \* h))) / (

1 + (cotq0 - (u[i] + h) \* Is \* vt / (P \* g \* yt0)) / tan(thth[i] + Kth3 \* h));

v[i + 1] = v[i] + (Kv1 + 2 \* Kv2 + 2 \* Kv3 + Kv4) \* h / 6;

y[i + 1] = y[i] + Is \* v[i] \* sin(thth[i]) / (P \* g) \* h;

x[i + 1] = x[i] + Is \* v[i] \* cos(thth[i]) / (P \* g) \* h;

thth[i + 1] = thth[i] + (Kth1 + 2 \* Kth2 + 2 \* Kth3 + Kth4) \* h / 6;

thtj[i + 1] = thth[i + 1] \* 180 / numpy.pi;

alph[i] = (v[i] \* (Kth1 + 2 \* Kth2 + 2 \* Kth3 + Kth4) / 6 + Is \* cos(thth[i]) / P) / (

p \* (v[i]) \*\* 2 \* cya[i] \* Is / (2 \* P \* p0 \* (1 - u[i])) + Is / (1 - u[i]));

alpj[i + 1] = alph[i] \* 180 / numpy.pi;

t[i + 1] = yt0 / vt \* (cotq0 - cotq);

xt[i + 1] = cotq \* yt0;

Drx[i + 1] = ((x[i + 1] - xt[i + 1]) \*\* 2 + (y[i + 1] - yt0) \*\* 2)\*\*0.5;

u[i + 1] = u[i] + h;

i = i + 1;

#作图

ShowResult.showResult(u,alpj,i,ma,alpha2,ma2,cy0,cya,yt0,t,y,thtj,v,Drx,xt,x)

1. Initialize.py文件代码：

import numpy

from scipy import interpolate

def initialize(yh,ah):

#导弹分离时参数

u=numpy.arange(1,500,0.1);#质量分数初始化

t=numpy.arange(1,500,0.1);#时间初始化

xt=numpy.arange(1,500,0.1);#目标x方位初始化

v=numpy.arange(1,500,0.1);#速度初始化

x=numpy.arange(1,500,0.1);#导弹x方位初始化

y=numpy.arange(1,500,0.1);#导弹y方位初始化

dr=numpy.arange(1,500,0.1);#弹幕距离初始化

cx=numpy.arange(1,500,0.1);

cya=numpy.arange(1,500,0.1);

alpj=numpy.arange(1,500,0.1);#俯仰角初始化（度）

alph=numpy.arange(1,500,0.1);#俯仰角初始化（弧）

thtj=numpy.arange(1,500,0.1);#弹道倾角初始化（度）

thth=numpy.arange(1,500,0.1);#弹道倾角初始化（弧）

v0=500;#初速度

t0=3;#初始时间

x0=674.0;#初始x方位

y0=329;#初始y方位

alp0=1.5;#初始alp角（度）

tht0=26;#初始tht角（度）

#动力系数

Is=2156;#发动机比冲

g=9.801;#重力加速度

P=2.2;#推重比

p0=5880;#翼载

#目标特性

vt=420;#目标速度

dt0=34200;#初始时刻弹目矩

yt0=15000;#目标y方向初始位置

xt0=numpy.sqrt(dt0\*\*2-(yt0-y0)\*\*2)+x0;#目标x方向初始位置

u[0] = 0; # 初始

t[0] = 0; # 初始时间

xt[0] = xt0; # 目标初始位置

cotq0 = xt0 / yt0; # 初视线角余切

v[0] = v0; # 导弹初始速度

x[0] = x0; # 导弹初始x

y[0] = y0; # 导弹初始y

dr[0] = ((x[0] - xt[0]) \*\* 2 + (y[0] - yt0) \*\* 2) \*\* 0.5; # 弹目初始距离

a = interpolate.interp1d(yh, ah)(y[0]); # 插值得到音速

alpj[0] = alp0; # 初始俯仰角（度）

alph[0] = alpj[0] / 180 \* numpy.pi; # 初始俯仰角（弧）

thtj[0] = tht0; # 初始弹道倾角[度]

thth[0] = thtj[0] / 180 \* numpy.pi; # 初始弹道倾角（弧）

return (u,t,xt,v,x,y,dr,cx,cya,alpj,alph,thtj,thth,v0,t0,x0,y0,alp0,tht0,Is,g,P,p0,vt,dt0,yt0,xt0,a,cotq0);

1. Interpolate.py文件：

import numpy

def interpolation():

# 阻力因数差值表

alpha1 = numpy.arange(2, 12, 2);

ma1 = [1.5, 2.1, 2.7, 3.3, 4.0];

cx0 = [

[0.0430, 0.0511, 0.0651, 0.0847, 0.1120],

[0.0360, 0.0436, 0.0558, 0.0736, 0.0973],

[0.0308, 0.0372, 0.0481, 0.0641, 0.0849],

[0.0265, 0.0323, 0.0419, 0.0560, 0.0746],

[0.0222, 0.0272, 0.0356, 0.0478, 0.0644]];

# 升力因数差值表

alpha2 = [1] + list(numpy.arange(2, 12, 2));

ma2 = numpy.arange(1.5, 4.5, 0.5);

cy0 = [

[0.0302, 0.0304, 0.0306, 0.0309, 0.0311, 0.0313],

[0.0279, 0.0280, 0.0284, 0.0286, 0.0288, 0.0290],

[0.0261, 0.0264, 0.0267, 0.0269, 0.0272, 0.0274],

[0.0247, 0.0248, 0.0251, 0.0254, 0.0257, 0.0259],

[0.0226, 0.0227, 0.0231, 0.0233, 0.0236, 0.0238],

[0.0209, 0.0210, 0.0213, 0.0216, 0.0219, 0.0221]];

# 大气密度

yh = numpy.arange(0, 23000, 1000);

ph = [1.22505, 1.11168, 1.00646, 0.90913, 0.81913, 0.73612,

0.65969, 0.58950, 0.52517, 0.46635, 0.41270, 0.36391,

0.31083, 0.26549, 0.22675, 0.19367, 0.16542, 0.14128,

0.12068, 0.10307, 0.08803, 0.07487, 0.06373];

ah = [340.29, 336.43, 332.53, 328.58, 324.58, 320.53, 316.43, 312.27,

308.06, 303.79, 299.46, 295.07, 295.07, 295.07, 295.07, 295.07,

295.07, 295.07, 295.07, 295.07, 295.07, 295.75, 296.43];

return (alpha1,ma1,cx0,alpha2,ma2,cy0,yh,ph,ah)

1. ShowResult.py文件：

from scipy import interpolate

import numpy

import matplotlib.pyplot as plt

plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签

plt.rcParams['axes.unicode\_minus']=False #用来正常显示负号

def showResult(u,alphaj,i,ma,alpha2,ma2,cy0,cya,yt0,t,y,thetaj,v,dr,xt,x):

print(' 计算得导弹的相对质量因数为:', u[i], '\n');

print(' 击中目标前导弹的攻角为:', alphaj[i], '\n');

CYA = interpolate.interp2d(alpha2, ma2, cy0, kind='linear')(alphaj[i], ma);

cya[i] = 180 / numpy.pi \* CYA;

yt = numpy.linspace(yt0, yt0, i);

plt.figure('导弹飞行高度随时间变化曲线')

plt.title('导弹飞行高度随时间变化曲线')

plt.plot(t[:i], y[:i])

plt.xlabel('t/s')

plt.ylabel('角度(°)')

plt.figure(' 迎角随时间变化曲线')

plt.title(' 迎角随时间变化曲线')

plt.plot(t[:i], alphaj[:i])

plt.xlabel("t/s")

plt.ylabel("角度(°)")

plt.figure(' 俯仰角随时间变化曲线')

plt.title(' 俯仰角随时间变化曲线')

plt.plot(t[:i], thetaj[:i] + alphaj[:i])

plt.xlabel('t/s')

plt.ylabel('角度(°)')

plt.figure(' 弹道倾角随时间变化曲线')

plt.plot(t[:i], thetaj[:i])

plt.title(' 弹道倾角随时间变化曲线')

plt.ylabel('角度(°)')

plt.xlabel('t/s')

plt.figure(' 导弹速度随时间变化曲线')

plt.plot(t[:i], v[:i])

plt.title(' 导弹速度随时间变化曲线')

plt.xlabel('t/s')

plt.ylabel('m/s')

plt.figure(' 单目斜距随时间变化曲线')

plt.plot(t[:i], dr[:i])

plt.title(' 单目斜距随时间变化曲线')

plt.xlabel('t/s')

plt.ylabel('m/s')

plt.figure(' 导弹和目标飞行高度随水平位置变化曲线');

plt.plot(x[:i], y[:i])

plt.plot(xt[:i], yt[:i])

for j in numpy.arange(0, 3600,600 ):

plt.plot((0, x[j]), (0, y[j]))

plt.plot((0, xt[j]), (0, yt[j]))

plt.title(' 导弹和目标飞行高度随水平位置变化曲线')

plt.xlabel('x/m')

plt.ylabel('y/m')

plt.figure('高度随相对质量因数变化曲线')

plt.plot(u[:i], y[:i])

plt.title('高度随相对质量因数变化曲线')

plt.xlabel('u')

plt.ylabel('角度(°)')

plt.figure(' 攻角随相对质量因数变化曲线')

plt.plot(u[:i], alphaj[:i])

plt.title(' 攻角随相对质量因数变化曲线')

plt.xlabel('u')

plt.ylabel(' 角度(°) ')

plt.figure(' 迎角随水平距离变化曲线')

plt.plot(x[:i], alphaj[:i])

plt.title(' 迎角随水平距离变化曲线')

plt.xlabel('x m')

plt.ylabel(' 角度(°) ')

plt.figure('相对质量因数随时间的变化曲线')

plt.title('相对质量因数随时间的变化曲线')

plt.plot(t[:i],u[:i])

plt.xlabel('t/s')

plt.ylabel('u')

plt.show()