**目录**

[一、 非线性系统与数值仿真基础（第一次上机） 2](#_Toc213783734)

[1.1 Ode45函数仿真 2](#_Toc213783735)

[1.2 Ode45函数仿真 5](#_Toc213783736)

[1.3 欧拉折线法仿真1.1 8](#_Toc213783737)

[二、 PID/PD控制 10](#_Toc213783738)

[2.1 PID控制 10](#_Toc213783739)

[2.2 PD控制 14](#_Toc213783740)

[2.3 三通道独立PD控制航天器三轴姿态 19](#_Toc213783741)

[三、 倒立摆系统控制（双通道PD+状态反馈） 23](#_Toc213783742)

[3.1 双通道PD控制（含力矩电机） 23](#_Toc213783743)

[3.2 状态反馈控制（无电机，place函数配置极点） 26](#_Toc213783744)

[四、 复杂干扰与运动轨迹（复合干扰控制+火箭飞行） 29](#_Toc213783745)

[4.1 符合干扰（常值+正弦）控制（增广系统+place） 29](#_Toc213783746)

[4.2 火箭增程弹飞行仿真（while循环+分段推力） 35](#_Toc213783747)

[五、 耦合系统与制导率（滑块摆锤+导弹制导） 39](#_Toc213783748)

[5.1 滑块——摆锤耦合系统非线性控制 39](#_Toc213783749)

[5.2 导弹制导率仿真（比例导引+升力限幅） 42](#_Toc213783750)

[六、 模板 47](#_Toc213783751)

[6.1 非线性系统与数值仿真基础 47](#_Toc213783752)

[6.1.1 ode45 函数仿真关键代码 47](#_Toc213783753)

[6.1.2 欧拉折线法仿真关键代码 47](#_Toc213783754)

[6.2 PID/PD控制设计模板 48](#_Toc213783755)

[6.2.1 PID控制仿真模板（以质点受扰运动为例） 48](#_Toc213783756)

[6.2.2 PD 控制关键代码（航天器姿态跟踪） 49](#_Toc213783757)

[6.3 倒立摆系统控制（双通道 PD + 状态反馈） 49](#_Toc213783758)

[6.3.1 双通道 PD 控制关键代码（含力矩电机） 49](#_Toc213783759)

[6.3.2 状态反馈控制关键代码（无电机，place 函数配置极点） 50](#_Toc213783760)

[6.4 复杂干扰与运动轨迹（复合干扰控制 + 火箭飞行） 51](#_Toc213783761)

[6.4.1 复合干扰（常值 + 正弦）控制关键代码（增广系统 + place） 51](#_Toc213783762)

[6.4.2 火箭增程弹飞行仿真关键代码（while 循环 + 分段推力） 52](#_Toc213783763)

[6.5 耦合系统与制导律（滑块摆锤 + 导弹制导） 53](#_Toc213783764)

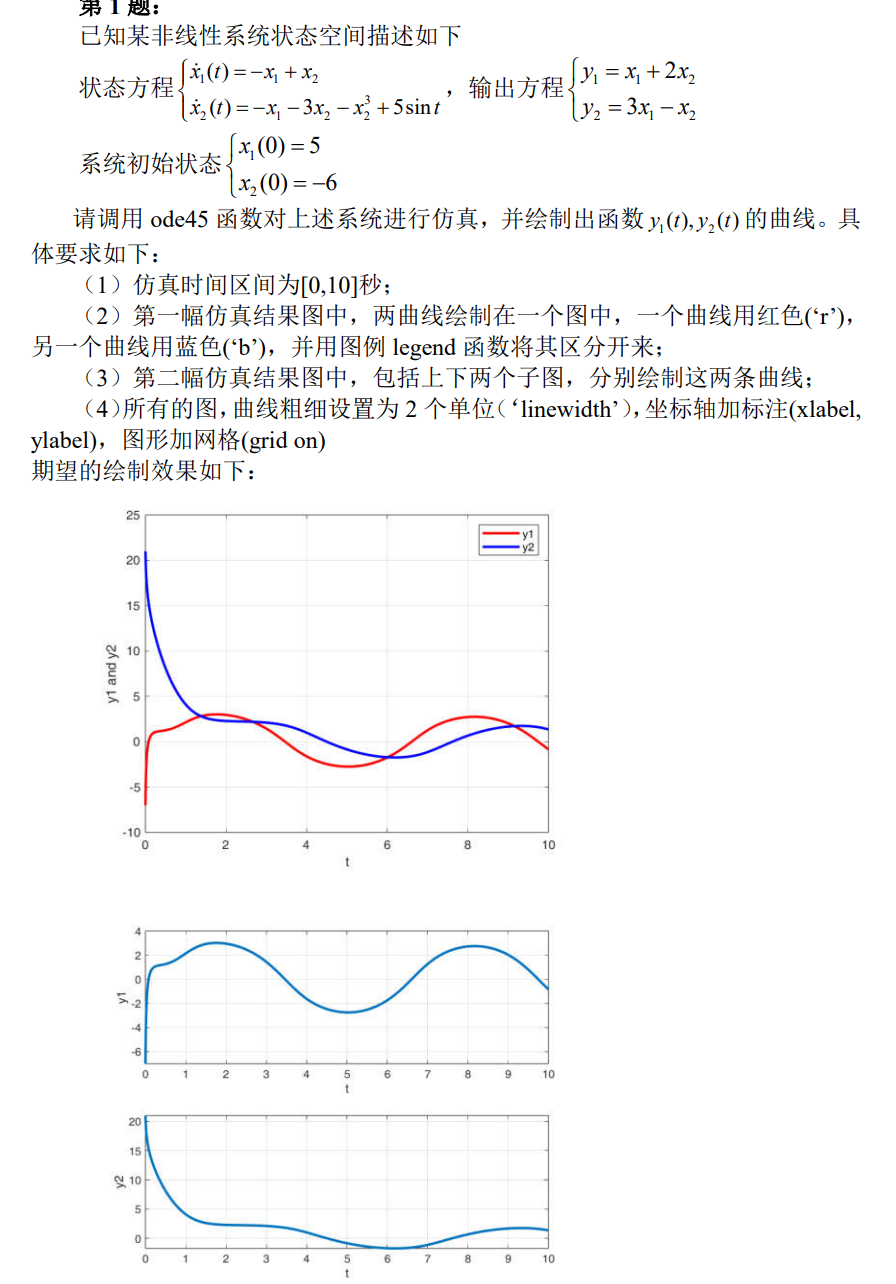
[6.5.1 滑块 - 摆锤耦合系统非线性控制关键代码 53](#_Toc213783765)

[6.5.2 导弹制导律仿真关键代码（比例导引 + 升力限幅） 54](#_Toc213783766)

[6.6 通用 55](#_Toc213783767)

# 非线性系统与数值仿真基础（第一次上机）

## Ode45函数仿真



%% 初始设置

close all,clear,clc,

tic,

% 初始化仿真时间区间

t0 = 0.0;

tf = 10.0;

%初始条件

x10 = 5;

x20 = -6;

state0 = [x10;x20]; %状态向量初值

%% 仿真

[tout,stateout] = ode45(@statequation01, [t0, tf], state0);

%输出t0-tf每一时刻的状态

x1out = stateout(:, 1);

x2out = stateout(:, 2);

y1out = x1out + 2\*x2out;

y2out = 3\*x1out - x2out;

%% 绘图

%画图1

figure(1);

plot(tout, y1out, "r", tout, y2out, "b", 'LineWidth', 2);

ylabel('y1 and y2');

xlabel('t');

grid on;

legend("y1","y2");

%画图2

figure(2);

subplot(2,1,1);

plot(tout, y1out, 'LineWidth', 2);

xlabel('t');ylabel('y1');

grid on;hold on;

subplot(2,1,2);

plot(tout, y2out, 'LineWidth', 2);

xlabel('t');ylabel('y2');

grid on;hold on;

toc

%% 函数

function statedot=statequation01(t,state)

% 函数

% x = [x1;x2]

x1 = state(1);

x2 = state(2);

%状态方程

u = 5\*sin(t);

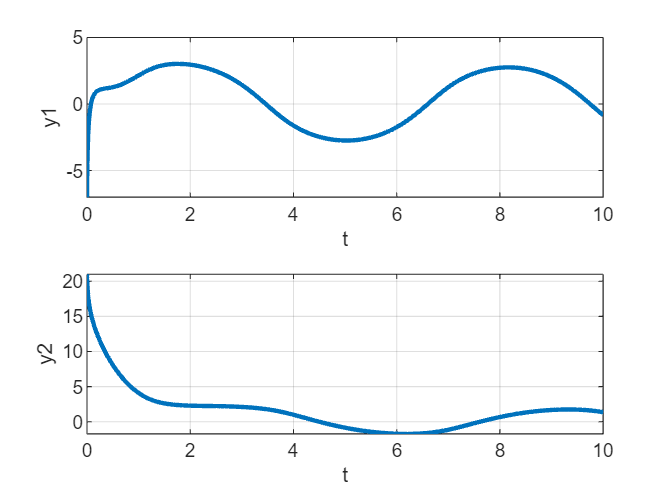
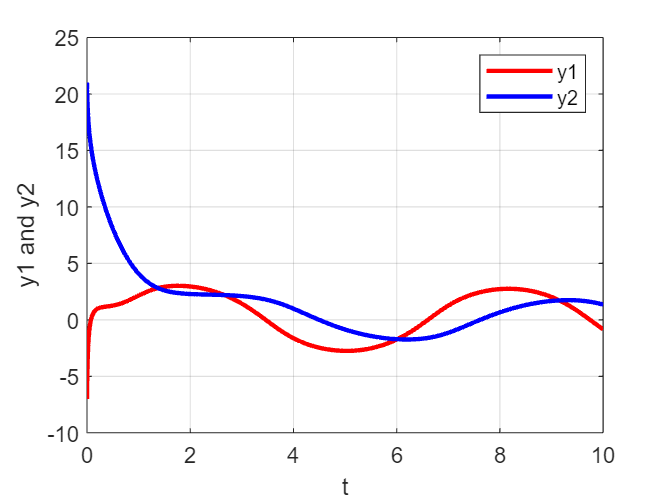
x1dot = -x1 + x2;

x2dot = -x1 - 3\*x2 - x2^3 + u;

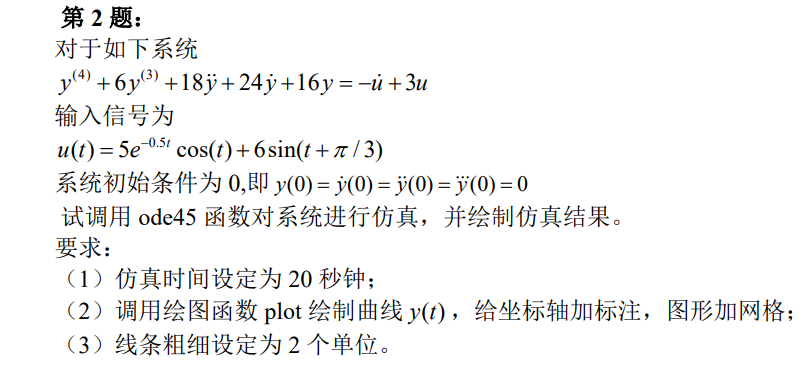
%输出得到的四个状态量

statedot=[x1dot;x2dot];

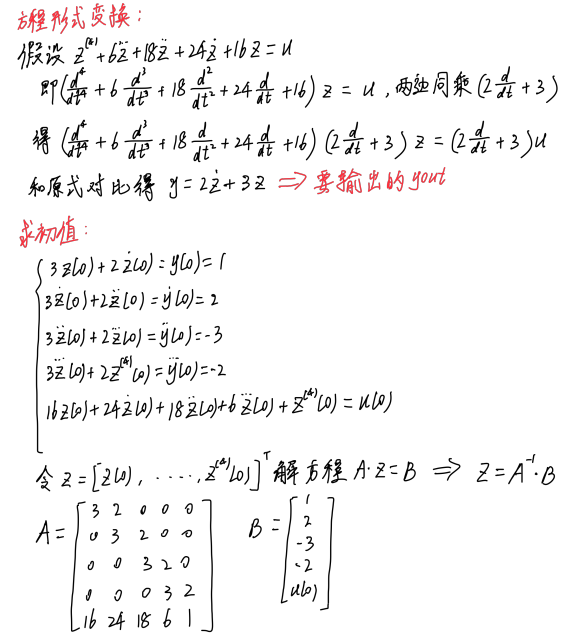
end

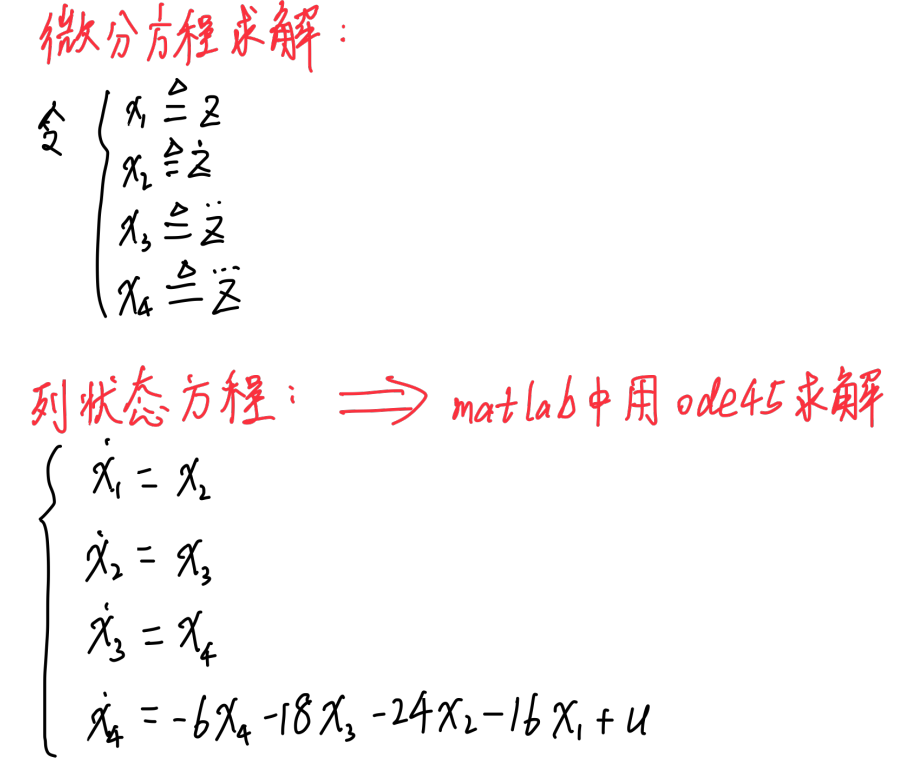


## Ode45函数仿真

****

**思路推导：**

****

****

%% 初始设置

close all,clear,clc,

tic,

% 初始化仿真时间区间

t0 = 0.0;

tf = 20.0;

%初始条件求解

A = [

3 -1 0 0 0;

0 3 -1 0 0;

0 0 3 -1 0;

0 0 0 3 -1;

16 24 18 6 1

];

B = [0;0;0;0;5\*exp(0)+6\*sin(pi/3)];

X0 = inv(A)\*B;

x10 = X0(1);

x20 = X0(2);

x30 = X0(3);

x40 = X0(4);

%% 仿真

state0 = [x10;x20;x30;x40]; %状态向量初值

[tout,stateout] = ode45(@statequation02, [t0, tf], state0);

%输出t0-tf每一时刻的状态

x1out = stateout(:, 1);

x2out = stateout(:, 2);

x3out = stateout(:, 3);

x4out = stateout(:, 4);

yout = 3\*x1out - x2out;

%画图

figure;

plot(tout, yout, 'LineWidth', 2);

ylabel('y');

xlabel('t');

grid on;

toc

function statedot=statequation02(t,state)

% 函数

% x = [y,y',y'',y''']

x1 = state(1);

x2 = state(2);

x3 = state(3);

x4 = state(4);

%状态方程

u = 5\*exp(-0.5\*t)\*cos(t) + 6\*sin(t+pi/3);

%du = -( 5\*0.5\*exp(-0.5\*t)\*cos(t) + 5\*exp(-0.5\*t)\*sin(t) ) + 6\*cos(t+pi/3);

x1dot = x2;

x2dot = x3;

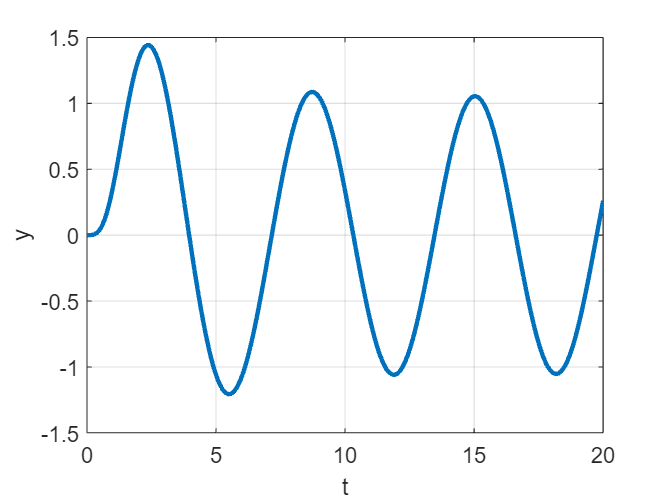
x3dot = x4;

x4dot = -6\*x4 - 18\*x3 - 24\*x2 - 16\*x1 + u;%3\*u - du;

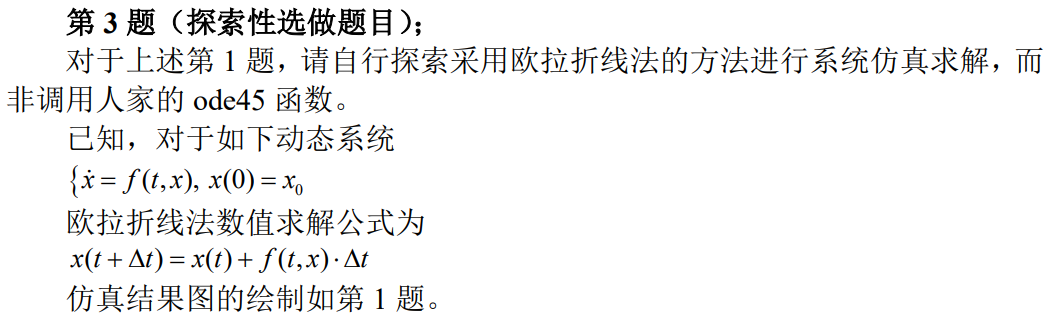
%输出得到的四个状态量

statedot=[x1dot;x2dot;x3dot;x4dot];

end



## 欧拉折线法仿真1.1

****

%% 初始设置

close all,clear,clc,

tic, %启动秒程序运行时间计时

% 初始化仿真时间区间

t0 = 0.0;

tf = 10.0;

%步长

t\_step = 0.0001;

tout = t0:t\_step:tf;

len = length(tout);

x1 = zeros(len, 1);

x2 = zeros(len, 1);

y1out = zeros(len, 1);

y2out = zeros(len, 1);

%初始条件

x10 = 5;

x20 = -6;

x1(1) = x10;

x2(1) = x20;

y1out(1) = x1(1) + 2\*x2(1);

y2out(1) = 3\*x1(1) - x2(1);

%% 仿真

%迭代求解

for t = 1:len-1

x2dot = -x1(t) - 3\*x2(t) - x2(t)\*x2(t)\*x2(t) + 5\*sin(tout(t));

x1dot = -x1(t) + x2(t);

x1(t+1) = x1(t) + x1dot \* t\_step;

x2(t+1) = x2(t) + x2dot \* t\_step;

y1out(t+1) = x1(t+1) + 2\*x2(t+1);

y2out(t+1) = 3\*x1(t+1) - x2(t+1);

end

%% 绘图

%画图1

figure(1);

plot(tout, y1out, "r", tout, y2out, "b", 'LineWidth', 2);

ylabel('y1 and y2');

xlabel('t');

grid on;

legend("y1","y2");

%画图2

figure(2);

subplot(2,1,1);

plot(tout, y1out, 'LineWidth', 2);

ylabel('y1');

xlabel('t');

grid on;

subplot(2,1,2);

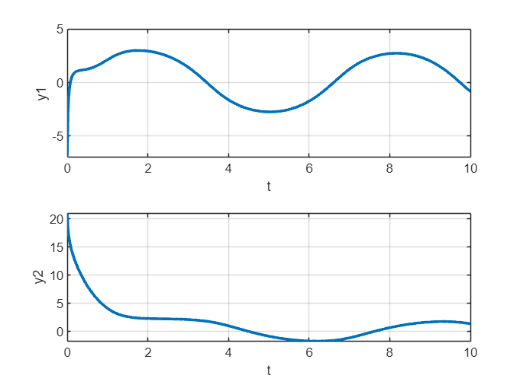
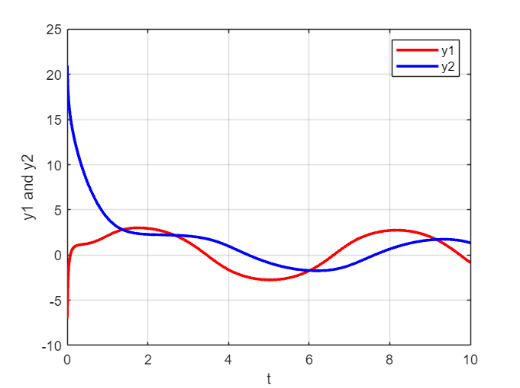
plot(tout, y2out, 'LineWidth', 2);

ylabel('y2');

xlabel('t');

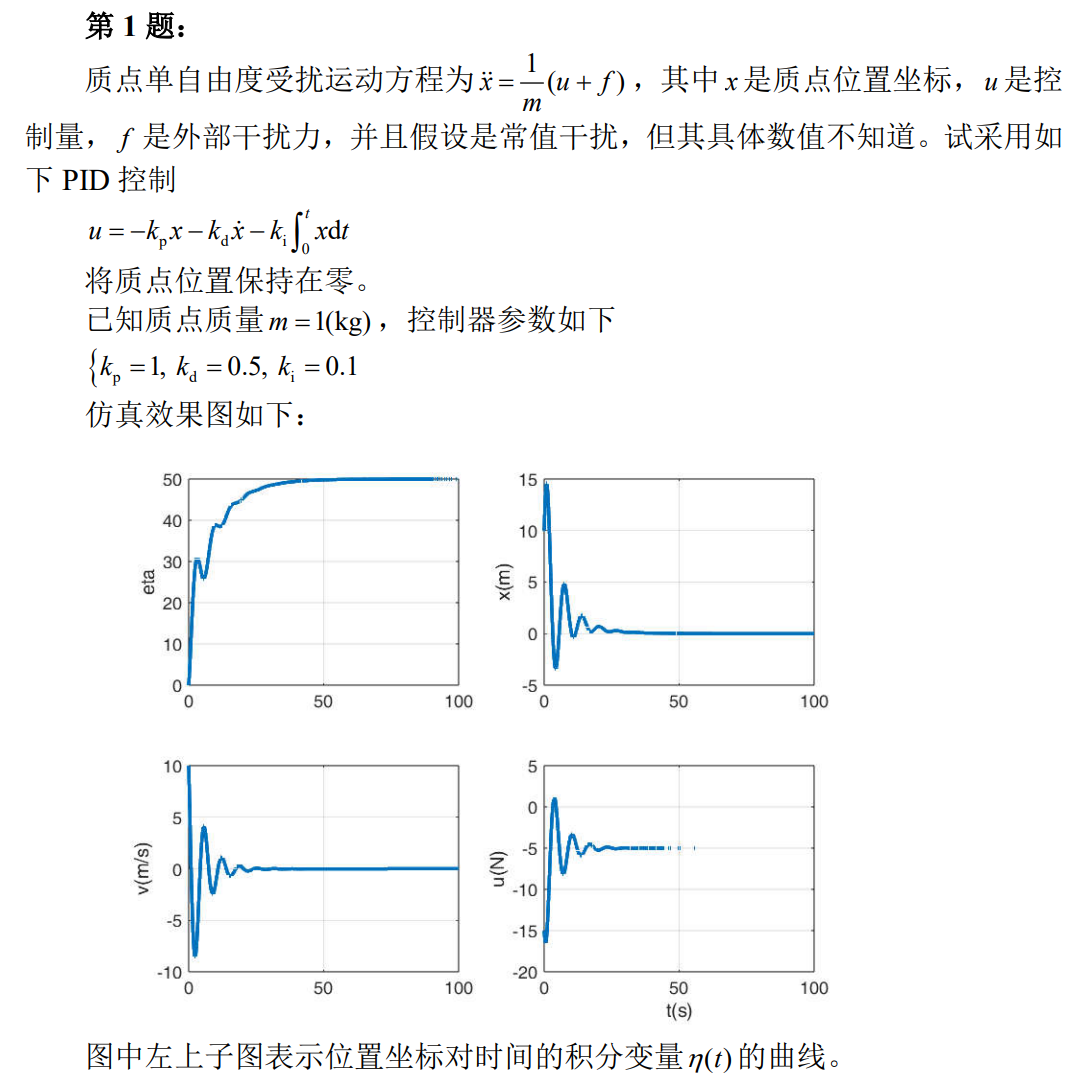
grid on;

toc %输出运行时间

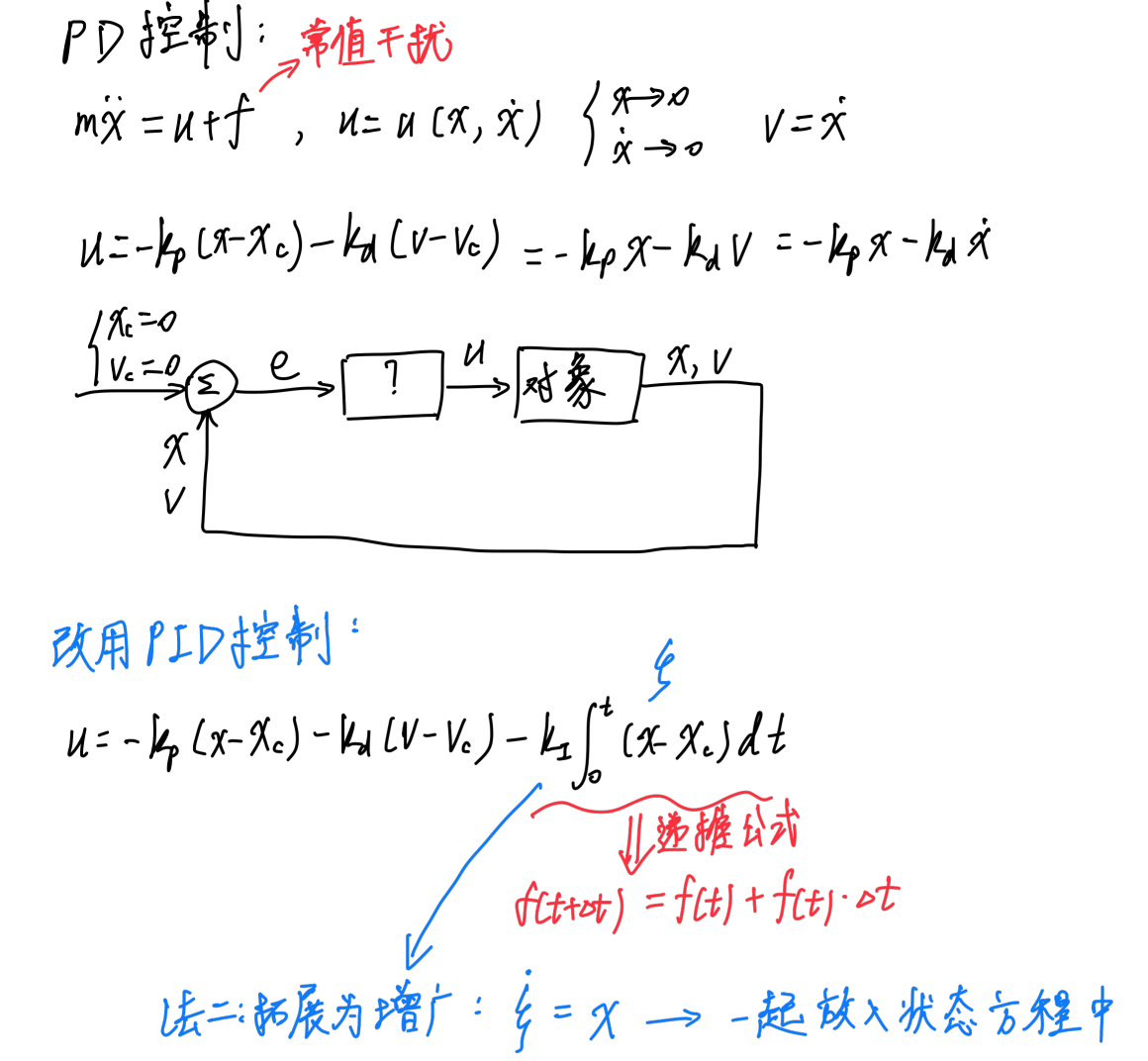


# PID/PD控制

## PID控制

****

**PD控制和PID控制思路：**

****

%% 初始设置

close all,clear,clc,

tic,

global m

m = 1;

kp = 1;

kd = 0.5;

ki = 0.1;

f = 5;

t0 = 0;

dt = 0.01;

tf = 100;

% 初始值

x = 10;

v = 10;

state = [x;v];

stateout = state;

uout = [];

tout = t0:dt:tf;

eta = 0;

etaout = eta;

xout = x;

vout = v;

%% 仿真

for t=t0:dt:tf

u = -kp \* x - kd \* v - ki \* eta;

uout = [uout, u];

% 迭代方程组

ke1 = statequation(t, state, u, f);

ke2 = statequation(t+0.5\*dt, state+0.5\*ke1\*dt, u, f);

ke3 = statequation(t+0.5\*dt, state+0.5\*ke2\*dt, u, f);

ke4 = statequation(t+dt, state+ke3\*dt, u, f);

state = state + 1/6\*(ke1 + 2\*ke2 + 2\*ke3 + ke4)\*dt;

stateout = [stateout, state];

xout = [xout, x];

vout = [vout, v];

x = state(1);

v = state(2);

eta = eta + x \* dt;

etaout = [etaout, eta];

end

tout = [tout, tf + dt];

uout = [uout, -kp \* x - kd \* v - ki \* eta];

%% 绘图

figure;

subplot(2, 2, 1);

hold on;

grid on;

plot(tout, etaout, 'LineWidth', 2);

ylabel('eta');

xlabel('t(s)');

subplot(2, 2, 2);

hold on;

grid on;

plot(tout, xout, 'LineWidth', 2);

ylabel('x(m)');

xlabel('t(s)');

subplot(2, 2, 3);

hold on;

grid on;

plot(tout, vout, 'LineWidth', 2);

ylabel('v(m/s)');

xlabel('t(s)');

subplot(2, 2, 4);

hold on;

grid on;

plot(tout, uout, 'LineWidth', 2);

ylabel('u(N)');

xlabel('t(s)');

toc,。

function output=statequation(t, state, u, f)

global m

x = state(1);

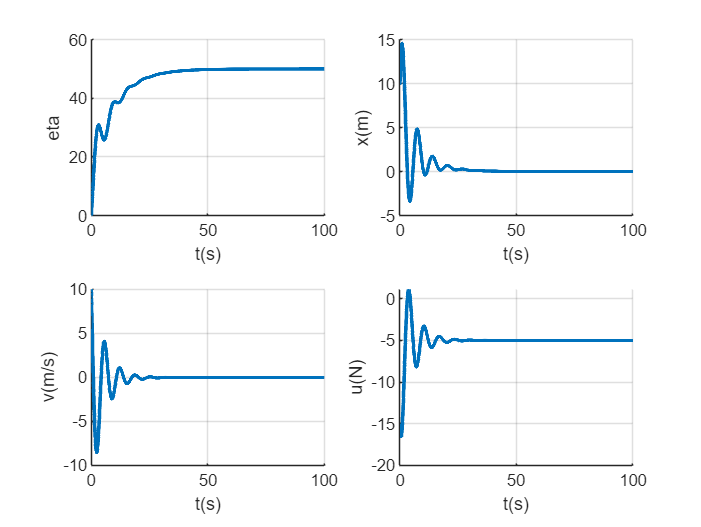
v = state(2);

xdot = v;

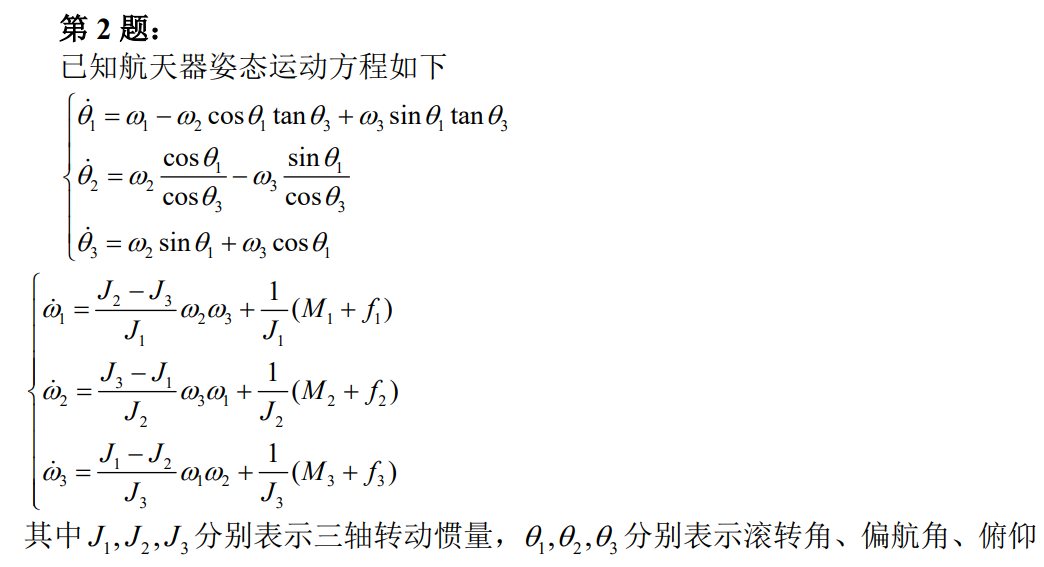
vdot = 1 / m \* (u + f);

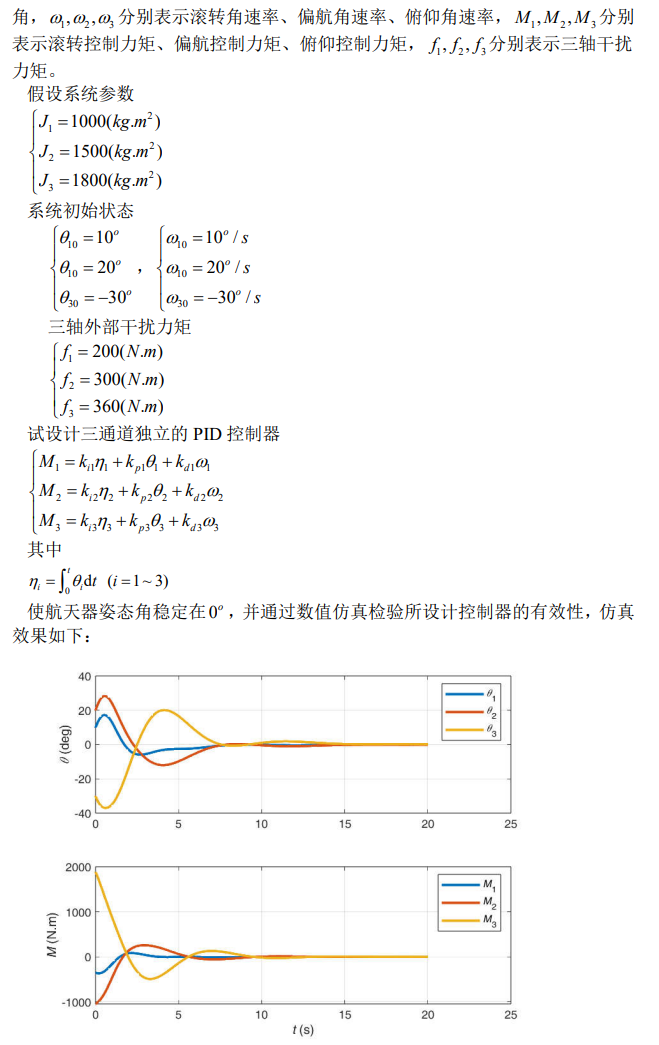
output = [xdot;vdot];

end

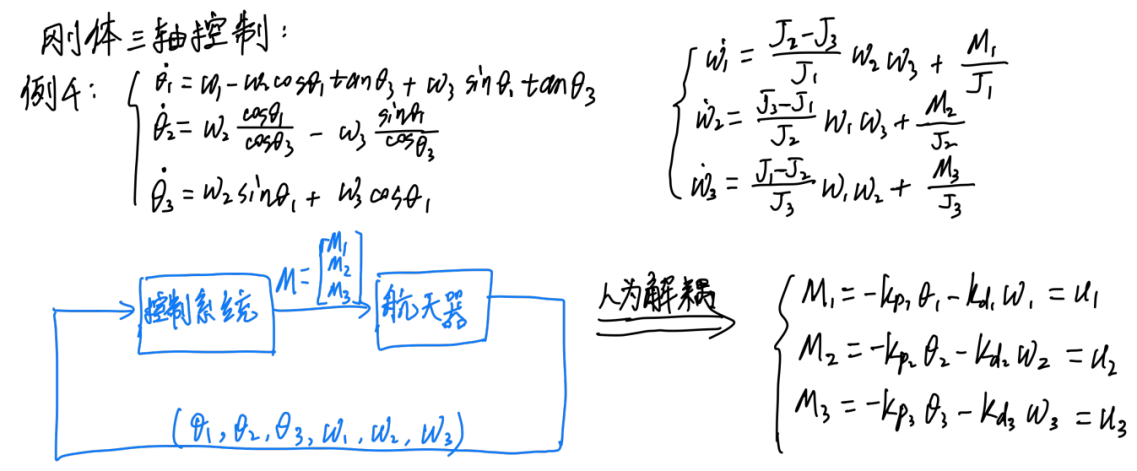


## PD控制

****

****

**思路：**

****

%% 初始参数设置

close all,clear,clc,

tic,

global J1 J2 J3 f1 f2 f3;

J1 = 1000; J2 = 1500; J3 = 1800;

f1 = 200; f2 = 300; f3 = 360;

% 初始化参数

t0 = 0.0;

dt = 0.1;

tf = 20.0;

tout = t0:dt:tf;

theta1 = deg2rad(10);

theta2 = deg2rad(20);

theta3 = deg2rad(-30);

omega1 = deg2rad(10);

omega2 = deg2rad(20);

omega3 = deg2rad(-30);

state = [theta1;theta2;theta3;omega1;omega2;omega3];

stateout = state;

uout = []; % 控制方程u

eta1 = 0;

eta2 = 0;

eta3 = 0;

theta1\_deg = 10;

theta2\_deg = 20;

theta3\_deg = -30;

theta1\_deg\_out = theta1\_deg;

theta2\_deg\_out = theta2\_deg;

theta3\_deg\_out = theta3\_deg;

% PID参数设置

kp1 = -1000; kd1 = -1000; ki1 = -300;%-0.1;

kp2 = -1500; kd2 = -1500; ki2 = -500;%-0.1;

kp3 = -1800; kd3 = -1800; ki3 = -500;%-0.1;

% 设置曲线标签

le\_str\_M = {"M\_1", "M\_2", "M\_3"};

le\_str\_theta = {"\theta\_1", "\theta\_2", "\theta\_3"};

%% 仿真

% 迭代

for t=t0:dt:tf

M1 = kp1\*theta1 + kd1\*omega1 + ki1\*eta1;

M2 = kp2\*theta2 + kd2\*omega2 + ki2\*eta2;

M3 = kp3\*theta3 + kd3\*omega3 + ki3\*eta3;

u = [M1;M2;M3];

uout = [uout, u];

% 迭代方程组

ke1 = statequation(t, state, u);

ke2 = statequation(t+0.5\*dt, state+0.5\*ke1\*dt, u);

ke3 = statequation(t+0.5\*dt, state+0.5\*ke2\*dt, u);

ke4 = statequation(t+dt, state+ke3\*dt, u);

state = state + 1/6\*(ke1 + 2\*ke2 + 2\*ke3 + ke4)\*dt;

stateout = [stateout, state];

% 状态变量更新

theta1 = state(1);

theta2 = state(2);

theta3 = state(3);

omega1 = state(4);

omega2 = state(5);

omega3 = state(6);

eta1 = eta1 + theta1 \* dt;

eta2 = eta2 + theta2 \* dt;

eta3 = eta3 + theta3 \* dt;

theta1\_deg = theta1 / pi \* 180;

theta2\_deg = theta2 / pi \* 180;

theta3\_deg = theta3 / pi \* 180;

theta1\_deg\_out = [theta1\_deg\_out, theta1\_deg];

theta2\_deg\_out = [theta2\_deg\_out, theta2\_deg];

theta3\_deg\_out = [theta3\_deg\_out, theta3\_deg];

end

theta1out = stateout(1, :);

theta2out = stateout(2, :);

theta3out = stateout(3, :);

omega1out = stateout(4, :);

omega2out = stateout(5, :);

omega3out = stateout(6, :);

% 循环最后一次更新

tout = [tout, tout(end)+dt];

M1 = kp1\*theta1 + kd1\*omega1 + ki1\*eta1;

M2 = kp2\*theta2 + kd2\*omega2 + ki2\*eta2;

M3 = kp3\*theta3 + kd3\*omega3 + ki3\*eta3;

u = [M1;M2;M3];

uout = [uout, u];

M1out = uout(1, :);

M2out = uout(2, :);

M3out = uout(3, :);

% 画图

figure(1);

hold on;grid on;

plot(tout, theta1\_deg\_out, 'LineWidth', 2);

plot(tout, theta2\_deg\_out, 'LineWidth', 2);

plot(tout, theta3\_deg\_out, 'LineWidth', 2);

xlabel('t(s)');ylabel('theta(deg)')

legend("\theta\_1", "\theta\_2", "\theta\_3");

figure(2);

hold on;grid on;

plot(tout, M1out, 'LineWidth', 2);

plot(tout, M2out, 'LineWidth', 2);

plot(tout, M3out, 'LineWidth', 2);

xlabel('t(s)');ylabel('M(N.m)');

legend("M\_1", "M\_2", "M\_3");

toc,

%% 函数

function output=statequation(t, state, u)

global J1 J2 J3 f1 f2 f3; % 转动惯量

% state = [theta1,theta2,theta3,w1,w2,w3]

% 获取此刻状态

theta1 = state(1);

theta2 = state(2);

theta3 = state(3);

omega1 = state(4);

omega2 = state(5);

omega3 = state(6);

M1 = u(1);

M2 = u(2);

M3 = u(3);

% 计算导数。作为状态输出

theta1dot = omega1 - omega2\*cos(theta1)\*tan(theta3) + omega3\*sin(theta1)\*tan(theta3);

theta2dot = omega2\*cos(theta1)/cos(theta3) - omega3\*sin(theta1)/cos(theta3);

theta3dot = omega2\*sin(theta1) + omega3\*cos(theta1);

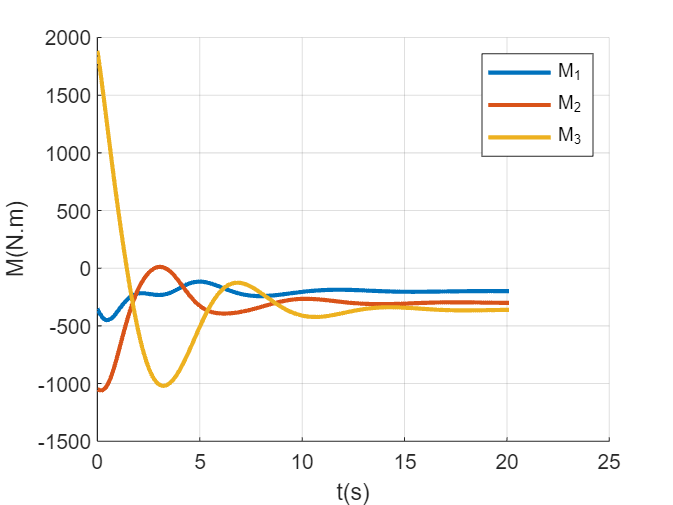
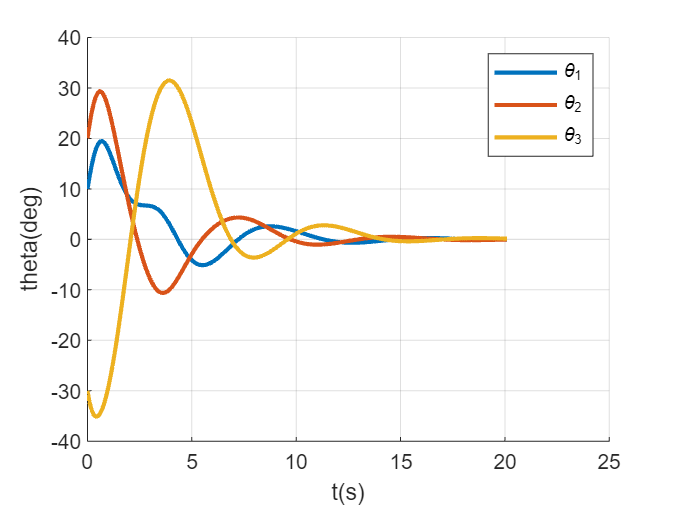
omega1dot = (J2-J3)/J1\*omega2\*omega3 + (M1+f1)/J1;

omega2dot = (J3-J1)/J2\*omega1\*omega3 + (M2+f2)/J2;

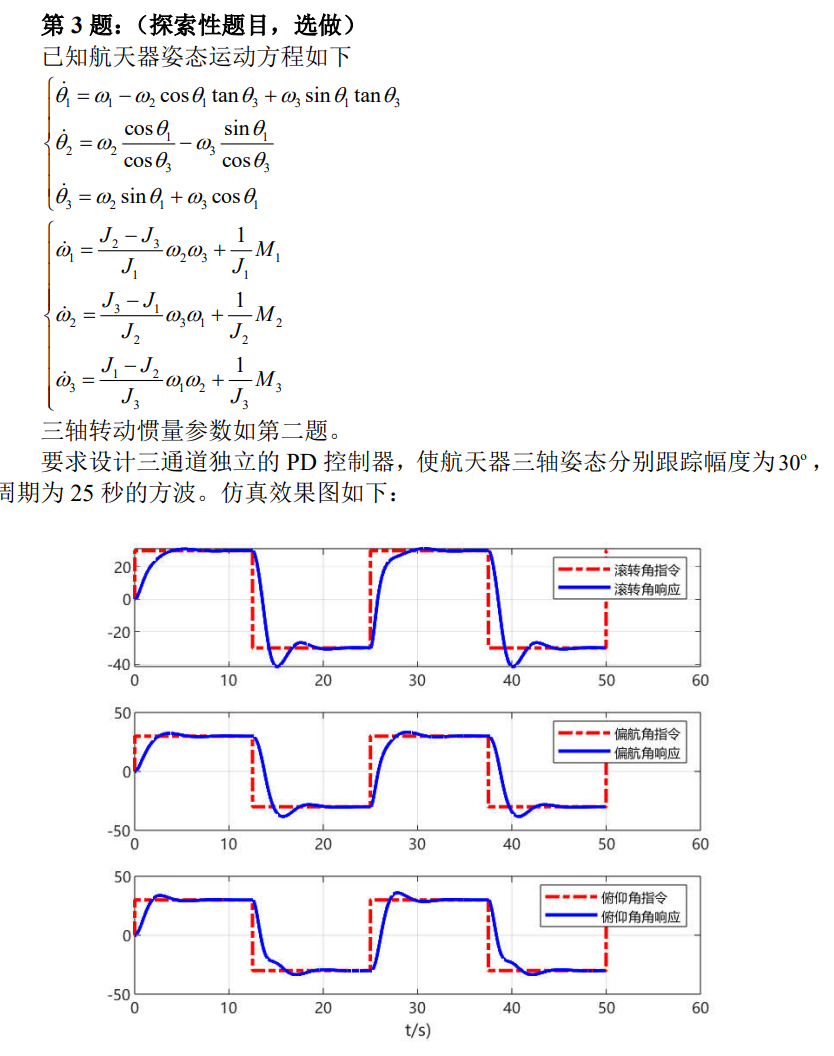
omega3dot = (J1-J2)/J3\*omega1\*omega2 + (M3+f3)/J3;

output = [theta1dot;theta2dot;theta3dot;omega1dot;omega2dot;omega3dot];

end



## 三通道独立PD控制航天器三轴姿态

****

%% 参数设置

close all,clear,clc,

tic,

global J1 J2 J3;

J1 = 1000; J2 = 1500; J3 = 1800;

% 初始化参数

t0 = 0.0;

dt = 0.01;

tf = 50.0;

tout = t0:dt:tf;

theta1 = deg2rad(0);

theta2 = deg2rad(0);

theta3 = deg2rad(0);

omega1 = deg2rad(0);

omega2 = deg2rad(0);

omega3 = deg2rad(0);

state = [theta1;theta2;theta3;omega1;omega2;omega3];

stateout = state;

uout = []; % 控制方程u

% PD参数设置

kp1 = -2000; kd1 = -2000;

kp2 = -3000; kd2 = -3000;

kp3 = -3600; kd3 = -3600;

%三轴方波周期

T1 = 25; T2 = 25; T3 = 25;

theta1cout = [];

theta2cout = [];

theta3cout = [];

%% 仿真

% 迭代

for t=t0:dt:tf

theta1c = 30\*pi/180\*sign(sin(2\*pi/T1\*t));

theta2c = 30\*pi/180\*sign(sin(2\*pi/T2\*t));

theta3c = 30\*pi/180\*sign(sin(2\*pi/T3\*t));

theta1cout = [theta1cout, theta1c];

theta2cout = [theta2cout, theta2c];

theta3cout = [theta3cout, theta3c];

error1 = theta1 - theta1c;

error2 = theta2 - theta2c;

error3 = theta3 - theta3c;

M1 = kp1\*error1 + kd1\*omega1; %跟随方波，所以改为误差值，其他控制不变

M2 = kp2\*error2 + kd2\*omega2;

M3 = kp3\*error3 + kd3\*omega3;

u = [M1;M2;M3];

uout = [uout, u];

% 迭代方程组

ke1 = statequation06(t, state, u);

ke2 = statequation06(t+0.5\*dt, state+0.5\*ke1\*dt, u);

ke3 = statequation06(t+0.5\*dt, state+0.5\*ke2\*dt, u);

ke4 = statequation06(t+dt, state+ke3\*dt, u);

state = state + 1/6\*(ke1 + 2\*ke2 + 2\*ke3 + ke4)\*dt;

stateout = [stateout, state];

% 状态变量更新

theta1 = state(1);

theta2 = state(2);

theta3 = state(3);

omega1 = state(4);

omega2 = state(5);

omega3 = state(6);

end

theta1out = stateout(1, :);

theta2out = stateout(2, :);

theta3out = stateout(3, :);

omega1out = stateout(4, :);

omega2out = stateout(5, :);

omega3out = stateout(6, :);

% 循环最后一次更新

tout = [tout, tout(end)+dt];

M1out = uout(1, :);

M2out = uout(2, :);

M3out = uout(3, :);

M1out = [M1out, M1out(end)];

M2out = [M2out, M2out(end)];

M3out = [M3out, M3out(end)];

theta1cout = [theta1cout, theta1cout(end)];

theta2cout = [theta2cout, theta2cout(end)];

theta3cout = [theta3cout, theta3cout(end)];

%% 绘图

% 画图

figure;

subplot(3, 1, 1),

plot(tout, theta1cout\*180/pi, 'r-.', tout, rad2deg(theta1out), 'b', 'LineWidth', 2);

set(gca, 'fontname', 'microsoft yahei');

legend('滚转角指令', '滚转角响应');

grid on;

subplot(3, 1, 2),

plot(tout, theta2cout\*180/pi, 'r-.', tout, rad2deg(theta2out), 'b', 'LineWidth', 2);

set(gca, 'fontname', 'microsoft yahei');

legend('偏航角指令', '偏航角响应');

grid on;

subplot(3, 1, 3),

plot(tout, theta3cout\*180/pi, 'r-.', tout, rad2deg(theta3out), 'b', 'LineWidth', 2);

set(gca, 'fontname', 'microsoft yahei');

legend('俯仰角指令', '俯仰角响应');

grid on;

toc,

%% 函数

function output=statequation06(t, state, u)

global J1 J2 J3; % 转动惯量

% state = [theta1;theta2;theta3;omega1;omega2;omega3];

% 获取此刻状态

theta1 = state(1);

theta2 = state(2);

theta3 = state(3);

omega1 = state(4);

omega2 = state(5);

omega3 = state(6);

M1 = u(1);

M2 = u(2);

M3 = u(3);

% 计算导数。作为状态输出

theta1dot = omega1 - omega2\*cos(theta1)\*tan(theta3) + omega3\*sin(theta1)\*tan(theta3);

theta2dot = omega2\*cos(theta1)/cos(theta3) - omega3\*sin(theta1)/cos(theta3);

theta3dot = omega2\*sin(theta1) + omega3\*cos(theta1);

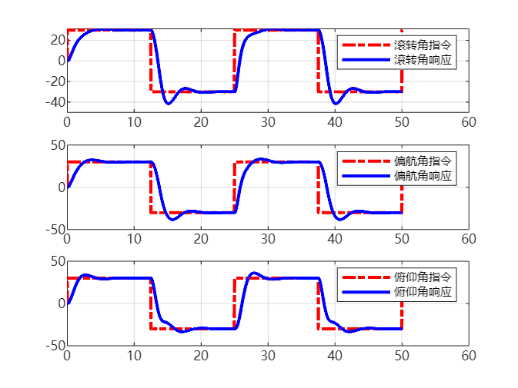
omega1dot = (J2-J3)/J1\*omega2\*omega3 + M1/J1;

omega2dot = (J3-J1)/J2\*omega1\*omega3 + M2/J2;

omega3dot = (J1-J2)/J3\*omega1\*omega2 + M3/J3;

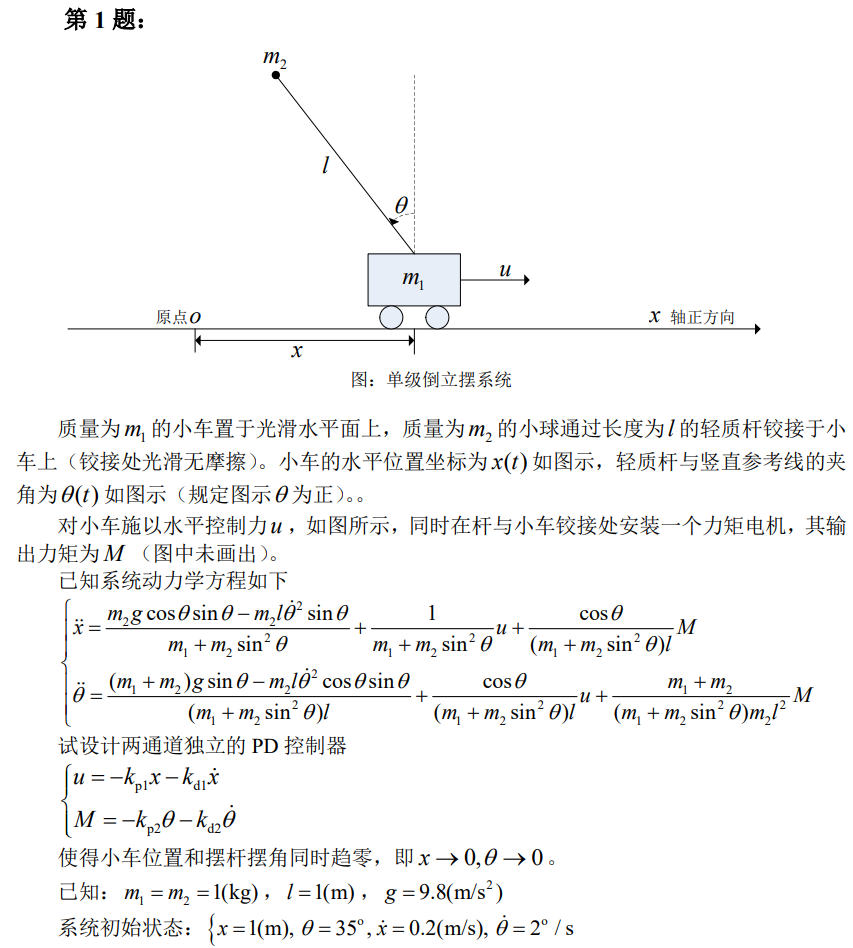
output = [theta1dot;theta2dot;theta3dot;omega1dot;omega2dot;omega3dot];

end



# 倒立摆系统控制（双通道PD+状态反馈）

## 双通道PD控制（含力矩电机）

****

%% 参数设置

close all,clear,clc,

tic,

global m1 m2 l g

m1 = 1;

m2 = 1;

l = 1;

g = 9.8;

x = 1;

theta = deg2rad(35);

v = 0.2;

omega = deg2rad(2);

state = [x;theta;v;omega];

stateout = state;

t0 = 0.0;

dt = 0.01;

tf = 10;

tout = t0:dt:tf;

uout = []; Mout = [];

kp1 = 10; kd1 = 10;

kp2 = 50; kd2 = 50;

%% 仿真

for t = t0:dt:tf

u = -kp1\*x - kd1\*v;

M = -kp2\*theta - kd2\*omega;

uout = [uout,u];

Mout = [Mout,M];

ke1 = stateequation(t, state, u, M);

ke2 = stateequation(t+0.5\*dt, state+0.5\*ke1\*dt, u, M);

ke3 = stateequation(t+0.5\*dt, state+0.5\*ke2\*dt, u, M);

ke4 = stateequation(t+dt, state+ke3\*dt, u, M);

state = state + 1/6\*(ke1 + 2\*ke2 + 2\*ke3 + ke4)\*dt;

stateout = [stateout, state];

x = state(1);

theta = state(2);

v = state(3);

omega = state(4);

end

xout = stateout(1, :);

thetaout = stateout(2, :);

vout = stateout(3, :);

omegaout = stateout(4, :);

uout = [uout,uout(end)];

Mout = [Mout,Mout(end)];

tout = [tout,tout(end)+dt];

%% 绘图

figure(1);

subplot(2, 1, 1);

hold on;grid on;

plot(tout, uout, 'LineWidth', 2);

xlabel('t');ylabel('u');

subplot(2, 1, 2);

hold on;grid on;

plot(tout, Mout, 'LineWidth', 2);

xlabel('t');ylabel('M');

figure(2);

hold on;grid on;

plot(tout, xout, 'LineWidth', 2);

plot(tout, thetaout, 'LineWidth', 2);

legend("x","\theta");

%% 函数

function statedot=stateequation(t, state, u, M)

global m1 m2 l g

% state = [x;theta;v;omega]

x = state(1);

theta = state(2);

v = state(3);

omega = state(4);

xdot = v;

thetadot = omega;

vdot = (m2\*g\*cos(theta)\*sin(theta) - m2\*l\*omega^2\*sin(theta))/(m1 + m2\*sin(theta)^2) + ...

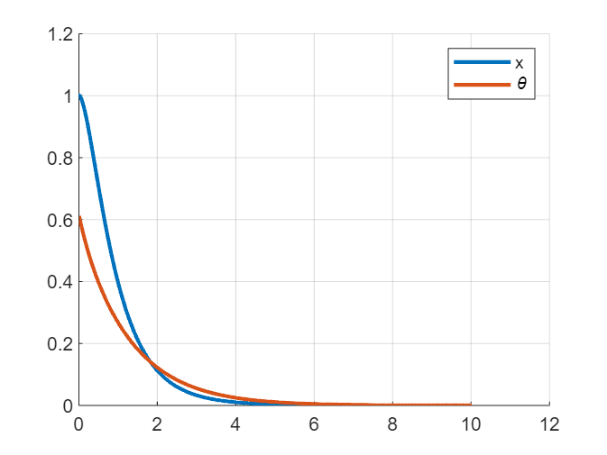
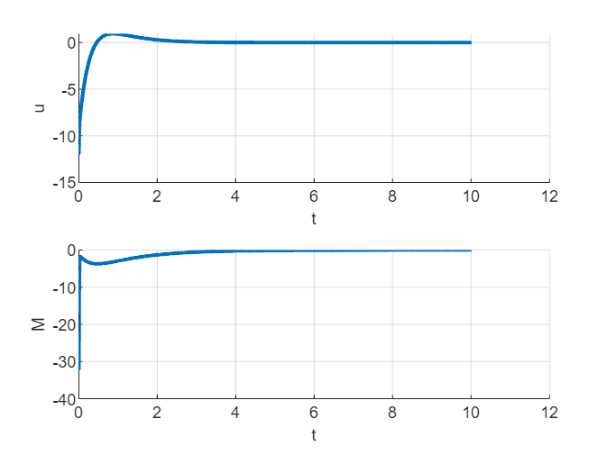
u/(m1 + m2\*sin(theta)^2) + cos(theta)\*M/(m1 + m2\*sin(theta)^2)/l;

omegadot = ((m1 + m2)\*g\*sin(theta) - m2\*l\*omega^2\*cos(theta)\*sin(theta))/(m1 + m2\*sin(theta)^2)/l + ...

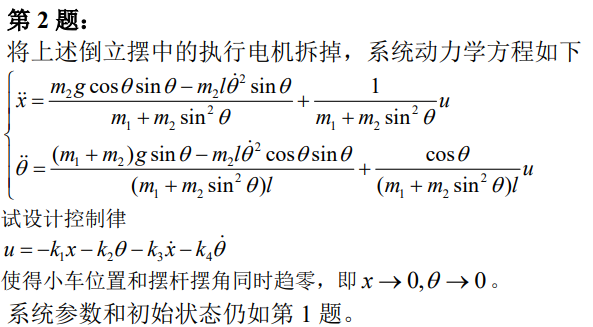
u\*cos(theta)/(m1 + m2\*sin(theta)^2)/l + (m1 + m2)\*M/(m1 + m2\*sin(theta)^2)/m2/l^2;

statedot = [xdot;thetadot;vdot;omegadot];

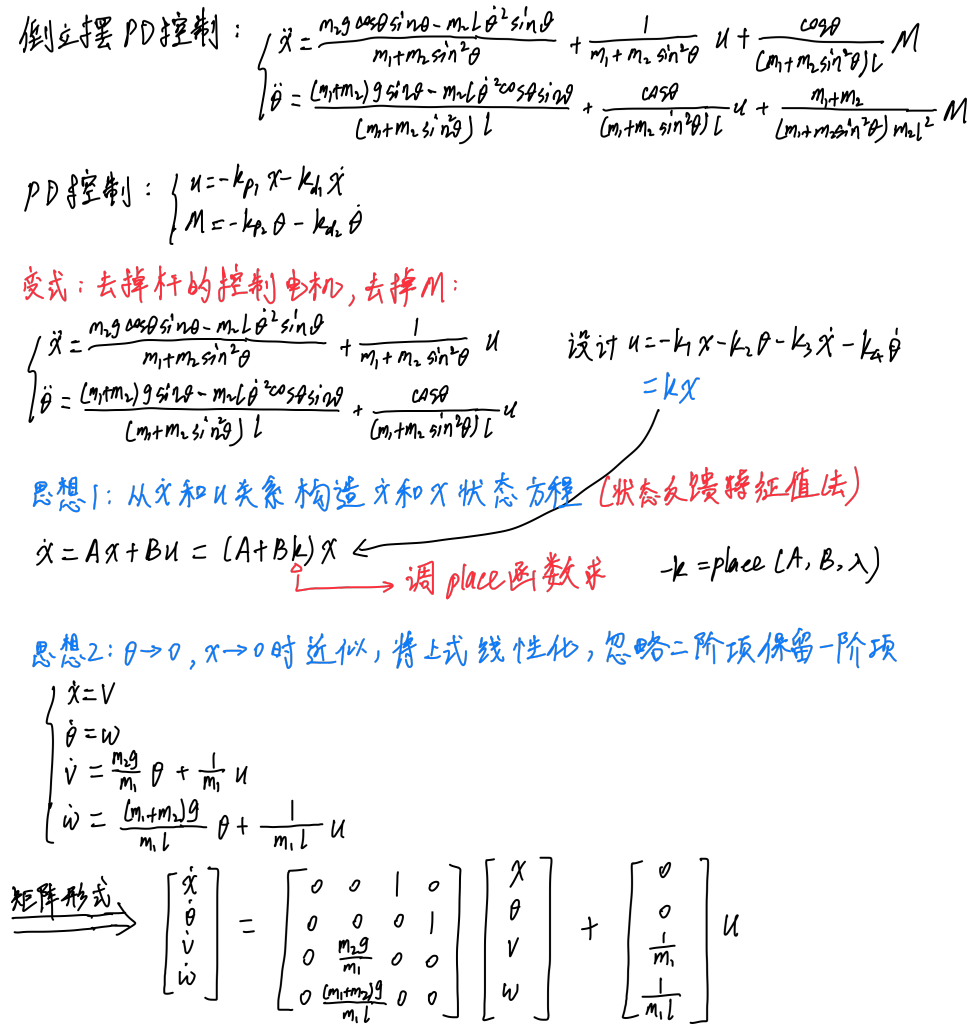
end



## 状态反馈控制（无电机，place函数配置极点）

****

**思路：**

****

%% 仿真初始设置

close all,clear,clc,

tic,

global m1 m2 l g

m1 = 1;

m2 = 1;

l = 1;

g = 9.8;

x = 1;

theta = deg2rad(35);

v = 0.2;

omega = deg2rad(2);

z = [x;theta;v;omega];

zout = z;

t0 = 0.0;

dt = 0.01;

tf = 10;

tout = t0:dt:tf;

uout = [];

% place直接求增益

A = [0 0 1 0; 0 0 0 1; 0 m2\*g/m1 0 0; 0 (m1+m2)\*g/m1/l 0 0];

B = [0; 0; 1/m1; 1/m1/l];

lambda = [-1+i; -1-i; -2+i; -2-i];

K = place(A, B, lambda); % 状态反馈增益矩阵

%% 仿真

for t = t0:dt:tf

u = -K\*z;

uout = [uout,u];

ke1 = stateequation(t, z, u);

ke2 = stateequation(t+0.5\*dt, z+0.5\*ke1\*dt, u);

ke3 = stateequation(t+0.5\*dt, z+0.5\*ke2\*dt, u);

ke4 = stateequation(t+dt, z+ke3\*dt, u);

z = z + 1/6\*(ke1 + 2\*ke2 + 2\*ke3 + ke4)\*dt;

zout = [zout, z];

x = z(1);

theta = z(2);

v = z(3);

omega = z(4);

end

xout = zout(1, :);

thetaout = zout(2, :);

vout = zout(3, :);

omegaout = zout(4, :);

uout = [uout,uout(end)];

tout = [tout,tout(end)+dt];

%% 绘图

figure;

subplot(2, 1, 1);

hold on;grid on;

plot(tout, uout, 'LineWidth', 2);

xlabel('t');ylabel('u');

subplot(2, 1, 2);

hold on;grid on;

plot(tout, xout, 'LineWidth', 2);

plot(tout, thetaout, 'LineWidth', 2);

xlabel('t');

legend("x","\theta");

%% 函数

function zdot=stateequation(t, z, u)

global m1 m2 l g

x = z(1);

theta = z(2);

v = z(3);

omega = z(4);

A = [m1 + m2, -m2\*l\*cos(theta); -m2\*l\*cos(theta), m2\*l^2];

B = [-m2\*l\*omega^2\*sin(theta) + u; m2\*l\*g\*sin(theta)];

xdot = v;

thetadot = omega;

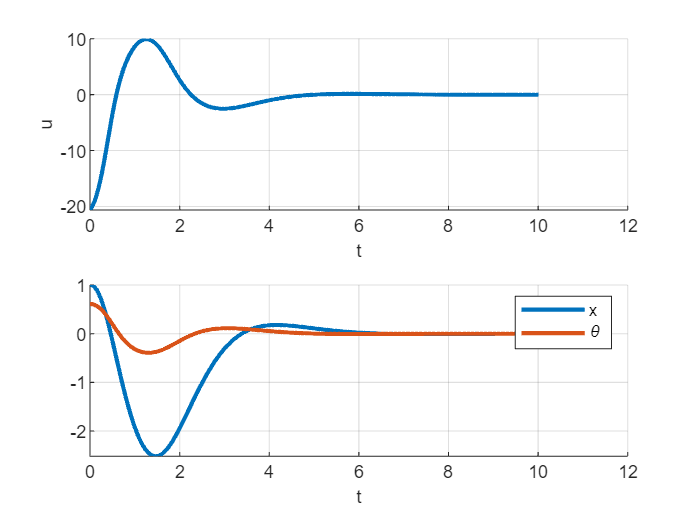
dotdot = inv(A)\*B;

vdot = dotdot(1);

omegadot = dotdot(2);

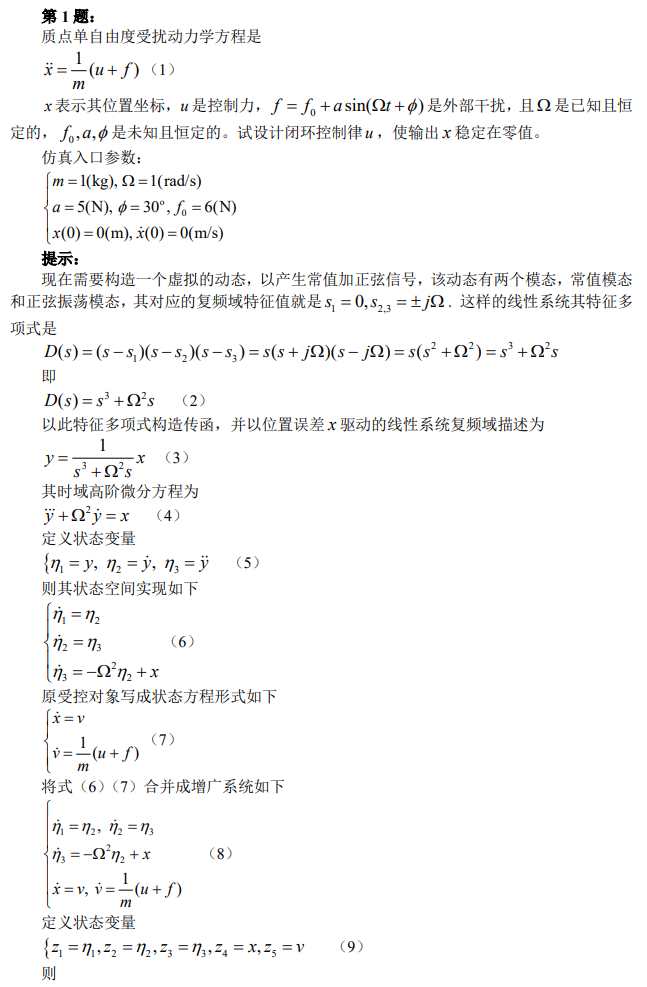
zdot = [xdot;thetadot;vdot;omegadot];

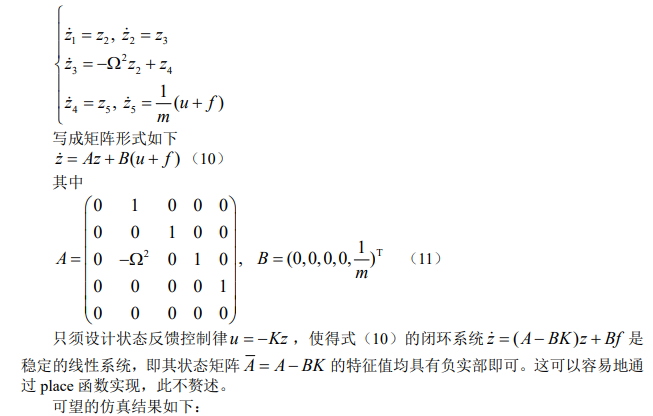
end

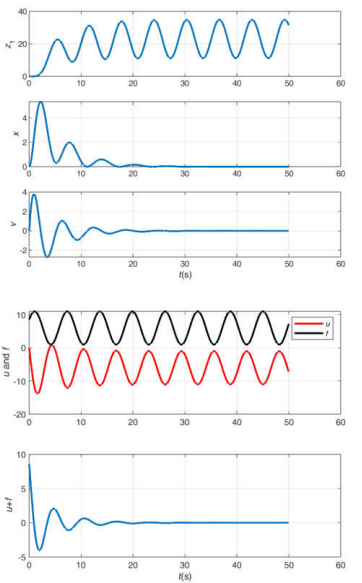
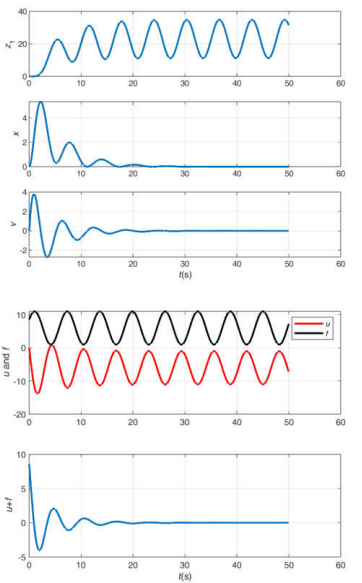


# 复杂干扰与运动轨迹（复合干扰控制+火箭飞行）

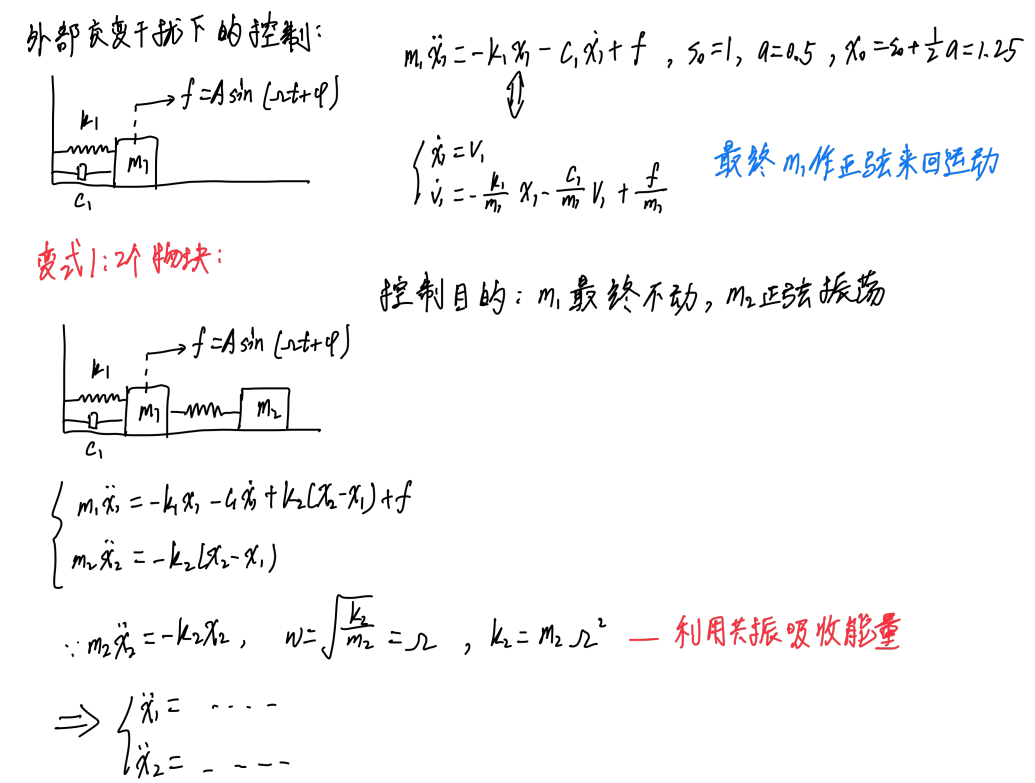
## 符合干扰（常值+正弦）控制（增广系统+place）

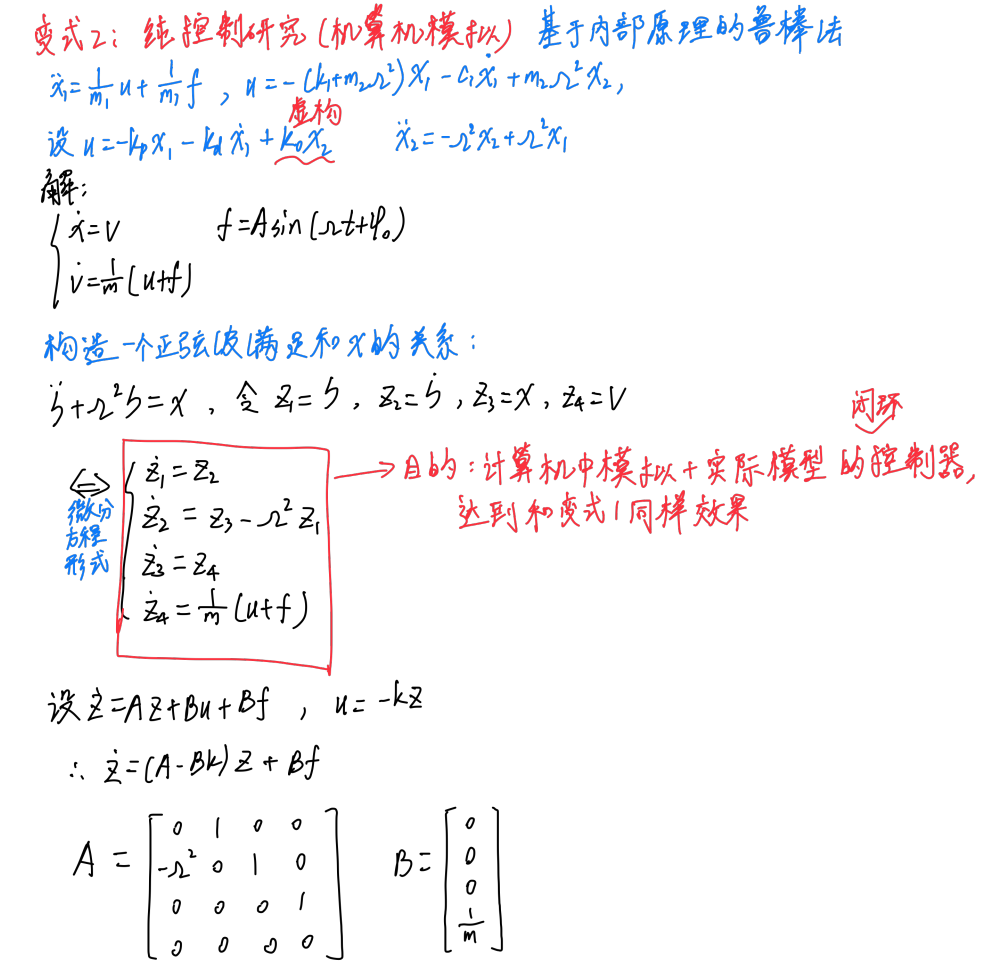
****

****

****

**思路：**

****

****

%% 参数设置

close all,clear,clc,

global m Omega

m = 1;

Omega = 1;

% 外部交变干扰

a = 5;

phi = 30\*pi/180;

f0 = 6;

t0 = 0.0;

dt = 0.01;

tf = 50;

tout = t0:dt:tf;

x0 = 0; %初始位置

v0 = 0; %初始速度

z1 = 0;

z2 = 0;

z3 = 0;

z4 = x0;

z5 = v0;

state = [z1;z2;z3;z4;z5];

stateout = state;

uout = [];

fout = [];

A = [0 1 0 0 0;

0 0 1 0 0;

0 -Omega^2 0 1 0;

0 0 0 0 1;

0 0 0 0 0];

B = [0; 0; 0; 0; 1/m];

lambda = [-0.2+i; -0.2-i; -0.5+i; -0.5-i; -0.25];

K = place(A, B, lambda);

%% 仿真

for t = t0:dt:tf

f = a\*sin(Omega\*t+phi) + f0;

fout = [fout, f];

u = -K\*state;

uout = [uout, u];

ke1 = stateequation(t, state, u, f);

ke2 = stateequation(t+0.5\*dt, state+0.5\*ke1\*dt, u, f);

ke3 = stateequation(t+0.5\*dt, state+0.5\*ke2\*dt, u, f);

ke4 = stateequation(t+dt, state+ke3\*dt, u, f);

state = state + 1/6\*(ke1 + 2\*ke2 + 2\*ke3 + ke4)\*dt;

stateout = [stateout, state];

end

z1out = stateout(1, :);

z2out = stateout(2, :);

z3out = stateout(3, :);

z4out = stateout(4, :);

z5out = stateout(5, :);

tout = [tout, tout(end)+dt];

fout = [fout, fout(end)];

uout = [uout, uout(end)];

%% 绘图

figure(1);

subplot(3,1,1);

hold on;grid on;

plot(tout, z1out, 'LineWidth', 2);

xlabel('t(s)');ylabel('z1');

subplot(3,1,2);

hold on;grid on;

plot(tout, z4out, 'LineWidth', 2);

xlabel('t(s)');ylabel('x');

subplot(3,1,3);

hold on;grid on;

plot(tout, z5out, 'LineWidth', 2);

xlabel('t(s)');ylabel('v');

figure(2);

subplot(2,1,1);

hold on;grid on;

plot(tout, uout, 'Color', 'red', 'LineWidth', 2);

plot(tout, fout, 'Color', 'black', 'LineWidth', 2);

xlabel('t(s)');ylabel('u and f');

legend("u","f")

subplot(2,1,2);

hold on;grid on;

plot(tout, uout + fout, 'LineWidth', 2);

xlabel('t(s)');ylabel('u+f');

function statedot=stateequation(t,state, u, f)

global m Omega

% state = [z1,z2,z3,z4,z5];

z1 = state(1);

z2 = state(2);

z3 = state(3);

z4 = state(4);

z5 = state(5);

%状态方程

z1dot = z2;

z2dot = z3;

z3dot = -Omega^2\*z2+z4;

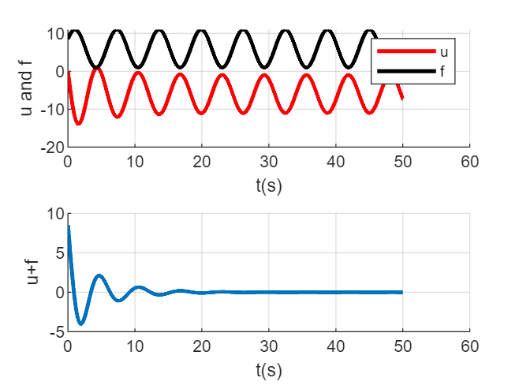
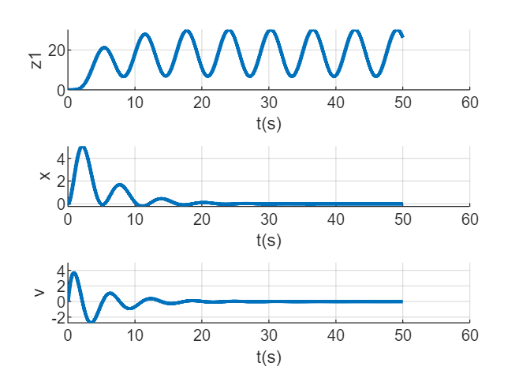
z4dot = z5;

z5dot = 1/m\*(u+f);

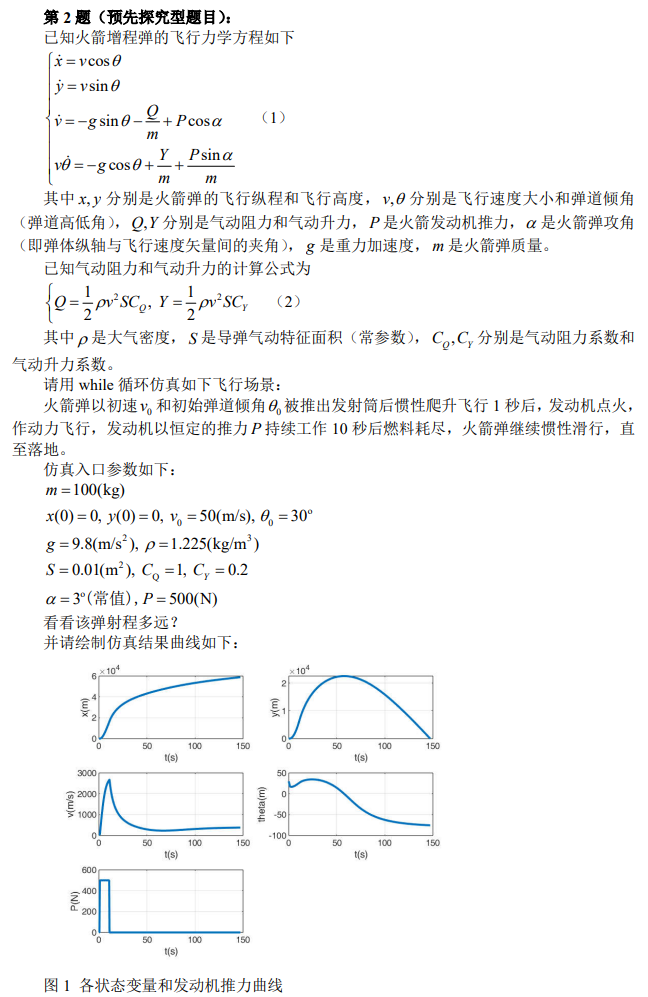
%输出得到的四个状态量

statedot=[z1dot;z2dot;z3dot;z4dot;z5dot];

end



## 火箭增程弹飞行仿真（while循环+分段推力）

****

%% 参数设置

close all,clear,clc,

global m g rho S Cq Cy alpha P

m = 100;

g = 9.8;

rho = 1.225;

S = 0.01;

Cq = 1;

Cy = 0.2;

alpha = 3/180\*pi;

P = 500;

v0 = 50;

theta0 = 30/180\*pi;

x = 0;

y = 0;

v = v0;

theta = theta0;

state = [x; y; v; theta0];

stateout = state;

dt = 0.005;

t = 0;

tout = t;

T = 10;

ton = 1; %力矩作用时间起始

toff = ton + T; %力矩作用时间结束

F = 500;

Pout = [];

%% 仿真

while y >= 0

if or(t < ton, t > toff) == 1

P = 0;

else

P = F;

end

Pout = [Pout, P];

ke1 = stateequation(t, state, P);

ke2 = stateequation(t+0.5\*dt, state+0.5\*ke1\*dt, P);

ke3 = stateequation(t+0.5\*dt, state+0.5\*ke2\*dt, P);

ke4 = stateequation(t+dt, state+ke3\*dt, P);

state = state + 1/6\*(ke1 + 2\*ke2 + 2\*ke3 + ke4)\*dt;

stateout = [stateout, state];

y = state(2);

t = t + dt;

tout = [tout, t];

end

xout = stateout(1, :);

yout = stateout(2, :);

vout = stateout(3, :);

thetaout = stateout(4, :);

Pout = [Pout, Pout(end)];

%% 绘图

figure(1);

hold on;grid on;

subplot(3,2,1);

plot(tout, xout, 'LineWidth', 2);

xlabel('t(s)');ylabel('x(m)');

subplot(3,2,2);

hold on;grid on;

plot(tout, yout, 'LineWidth', 2);

xlabel('t(s)');ylabel('y(m)');

subplot(3,2,3);

hold on;grid on;

plot(tout, vout, 'LineWidth', 2);

xlabel('t(s)');ylabel('v(m/s)');

subplot(3,2,4);

hold on;grid on;

plot(tout, rad2deg(thetaout), 'LineWidth', 2);

xlabel('t(s)');ylabel('theta(m)');

grid on;

subplot(3,2,5);

hold on;grid on;

plot(tout, Pout, 'LineWidth', 2);

xlabel('t(s)');ylabel('P(N)');

figure(2);

hold on;grid on;

plot(xout,yout,"LineWidth",2);

xlabel('x');ylabel('y');

function statedot=stateequation(t,state, P)

global m g rho S Cq Cy alpha

% state = [x,y,v,theta];

x = state(1);

y = state(2);

v = state(3);

theta = state(4);

xdot = v\*cos(theta);

ydot = v\*sin(theta);

Q = 0.5\*rho\*v^2\*S\*Cq;

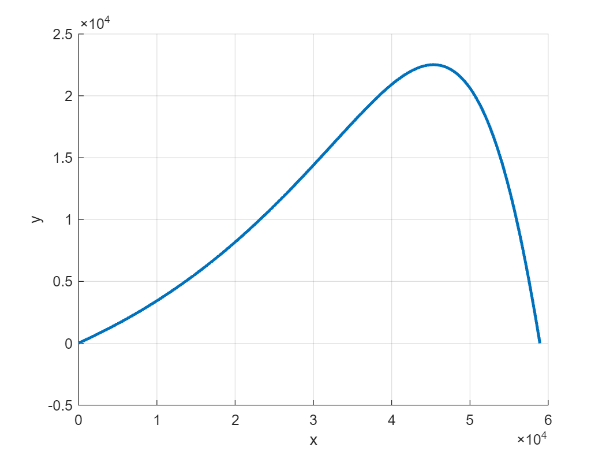
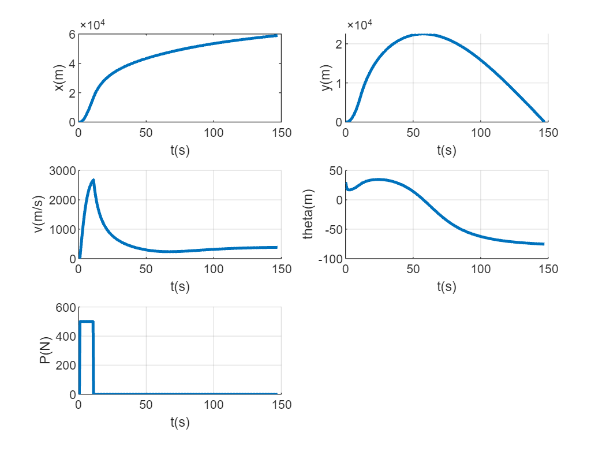
Y = 0.5\*rho\*v^2\*S\*Cy;

vdot = -g\*sin(theta) - Q/m + P\*cos(alpha);

thetadot = 1/v\*(-g\*cos(theta) + Y/m + P\*sin(alpha)/m);

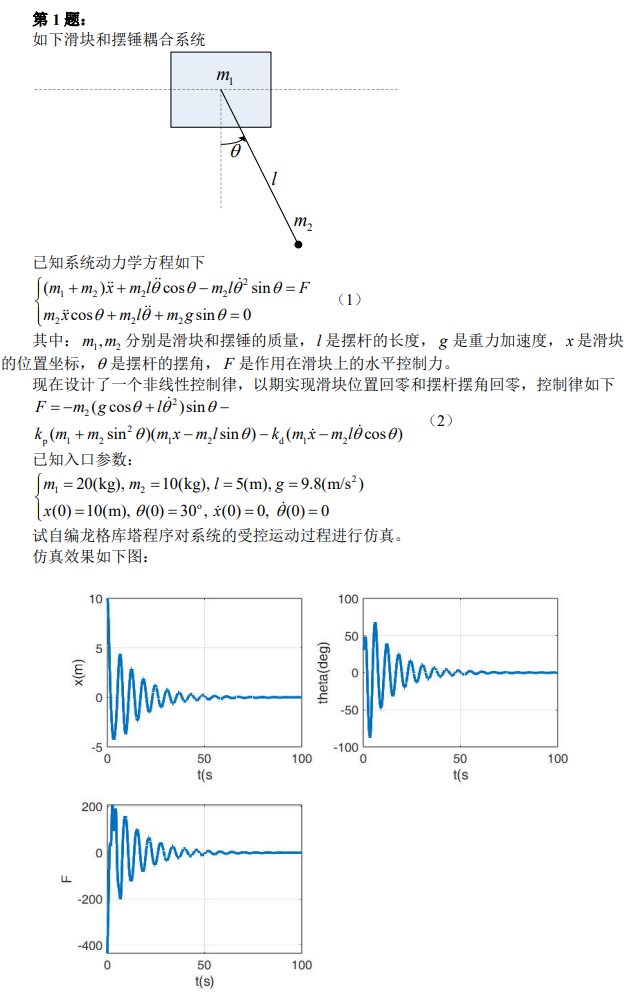
statedot = [xdot; ydot; vdot; thetadot];

end



# 耦合系统与制导率（滑块摆锤+导弹制导）

## 滑块——摆锤耦合系统非线性控制

****

close all,clear,clc,

tic,

%% 参数设置

global m1 m2 l g

m1 = 20;

m2 = 10;

l = 5;

g = 9.8;

kp = 0.1;

kd = 1;

x = 10;

theta = 30/180\*pi;

v = 0;

omega = 0;

state = [x;theta;v;omega];

stateout = state;

t0 = 0.0;

dt = 0.01;

tf = 100;

tout = t0:dt:tf;

Fout = [];

%% 仿真

for t = t0:dt:tf

F = -m2\*(g\*cos(theta) + l\*omega^2)\*sin(theta) - kp\*(m1 + ...

m2\*sin(theta)\*sin(theta))\*(m1\*x - m2\*l\*sin(theta)) - ...

kd\*(m1\*v - m2\*l\*omega\*cos(theta));

Fout = [Fout,F];

ke1 = stateequation(t, state, F);

ke2 = stateequation(t+0.5\*dt, state+0.5\*ke1\*dt, F);

ke3 = stateequation(t+0.5\*dt, state+0.5\*ke2\*dt, F);

ke4 = stateequation(t+dt, state+ke3\*dt, F);

state = state + 1/6\*(ke1 + 2\*ke2 + 2\*ke3 + ke4)\*dt;

stateout = [stateout, state];

x = state(1);

theta = state(2);

v = state(3);

omega = state(4);

end

xout = stateout(1, :);

thetaout = stateout(2, :);

vout = stateout(3, :);

omegaout = stateout(4, :);

Fout = [Fout,Fout(end)];

tout = [tout,tout(end)+dt];

%% 绘图

figure;

subplot(2, 2, 1);

plot(tout, xout, 'LineWidth', 2);

xlabel('t(s)');ylabel('x(m)');

hold on;grid on;

subplot(2, 2, 2);

plot(tout, thetaout/pi\*180, 'LineWidth', 2);

xlabel('t(s)');ylabel('theta(deg)');

hold on;grid on;

subplot(2, 2, 3);

plot(tout, Fout, 'LineWidth', 2);

xlabel('t(s)');ylabel('F');

hold on;grid on;

function statedot=stateequation(t, state, F)

global m1 m2 l g

%state = [x;theta;v;omega];

x = state(1);

theta = state(2);

v = state(3);

omega = state(4);

xdot = v;

thetadot = omega;

vdot = (F + m2\*l\*omega^2\*sin(theta) + ...

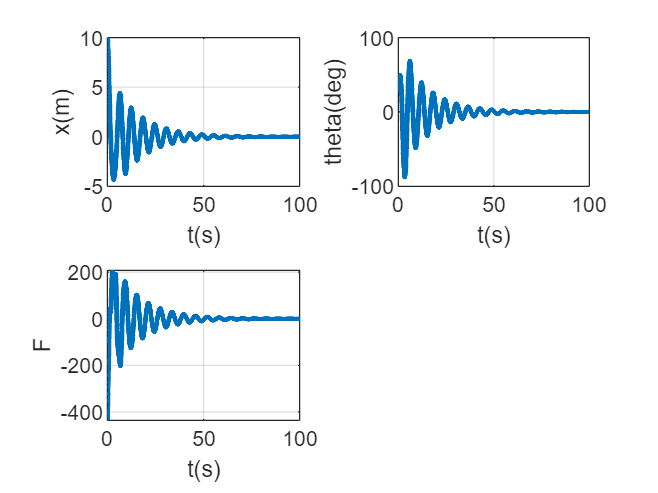
m2\*g\*sin(theta)\*cos(theta))/(m1 + m2\*sin(theta)\*sin(theta));

omegadot = -(F\*cos(theta) + m2\*l\*omega^2\*sin(theta)\*cos(theta) + ...

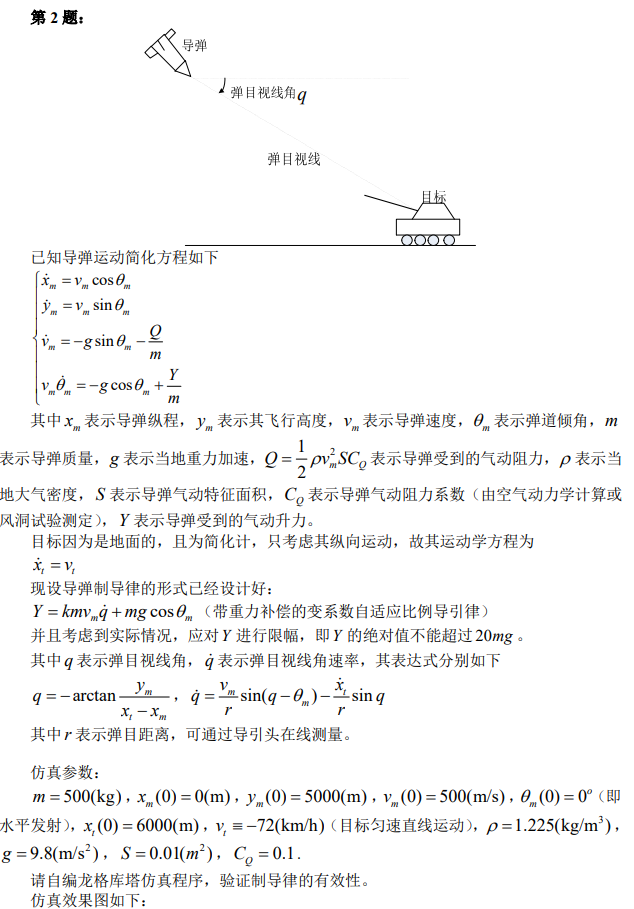
(m1 + m2)\*g\*sin(theta))/(m1 + m2\*sin(theta)\*sin(theta))/l;

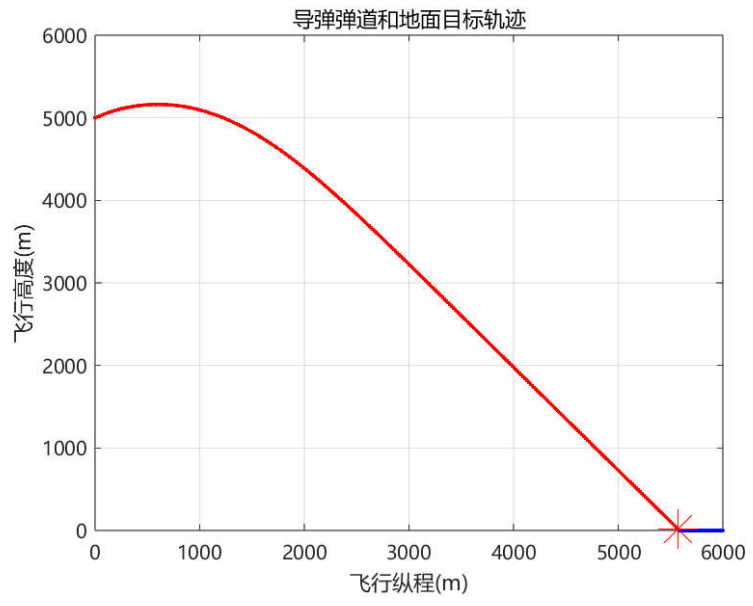
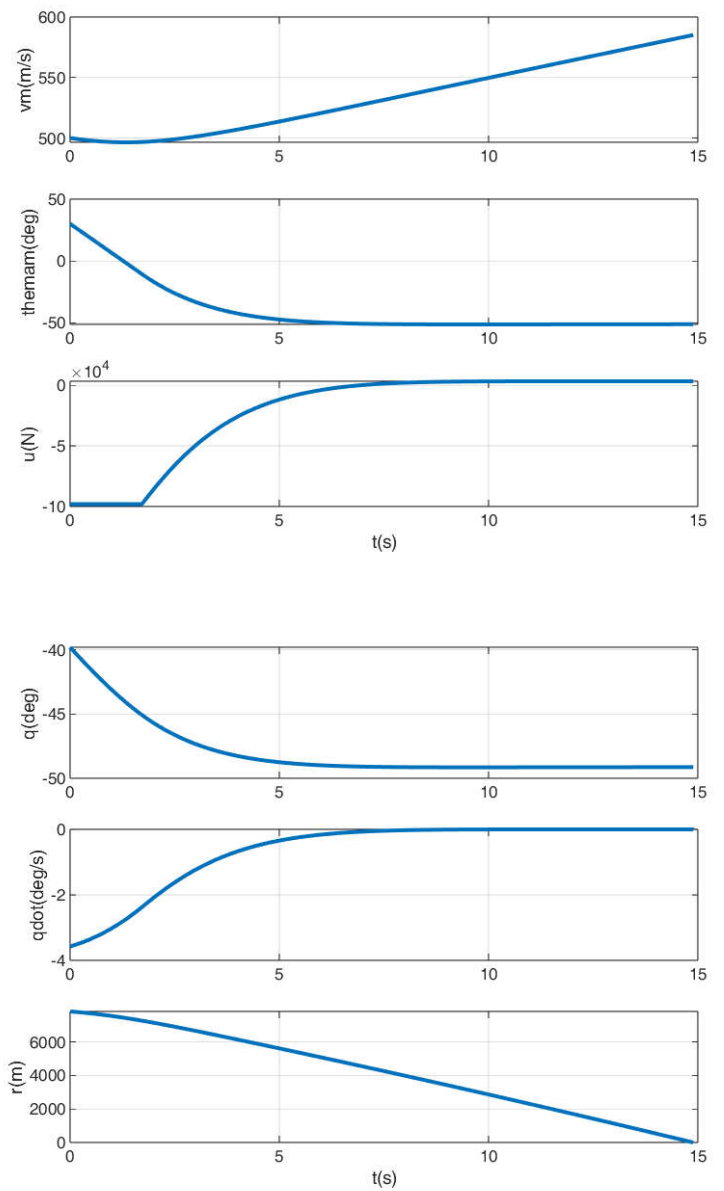
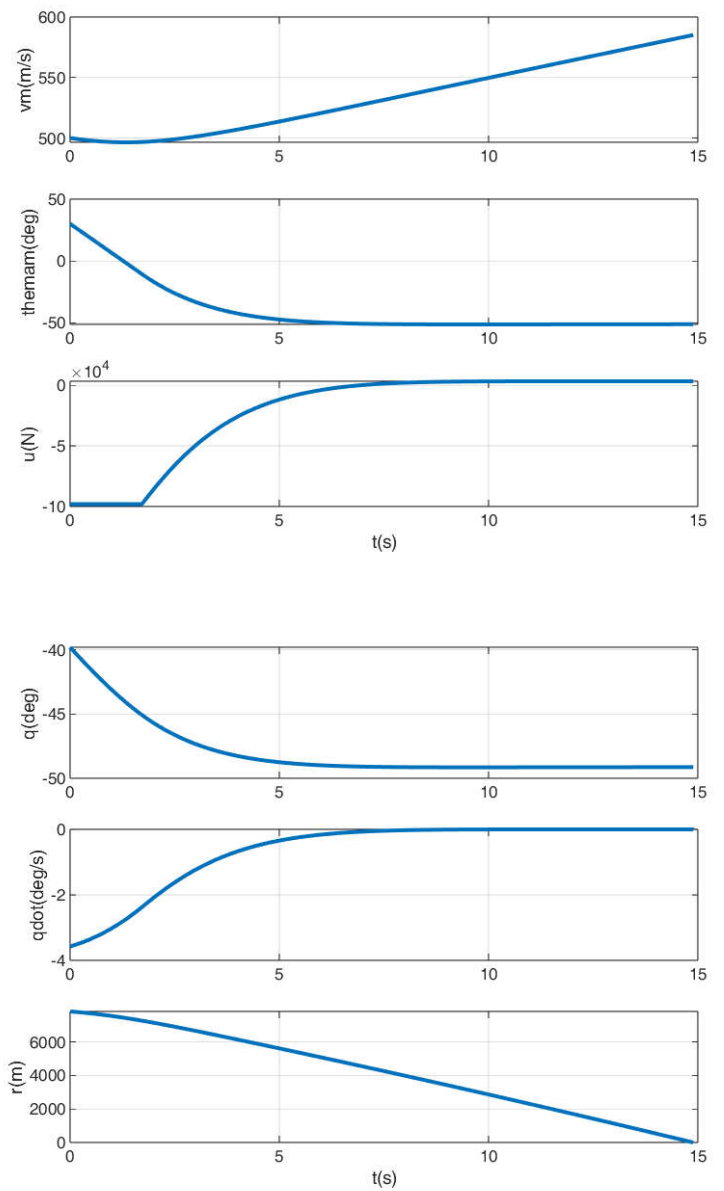
statedot = [xdot;thetadot;vdot;omegadot];

end

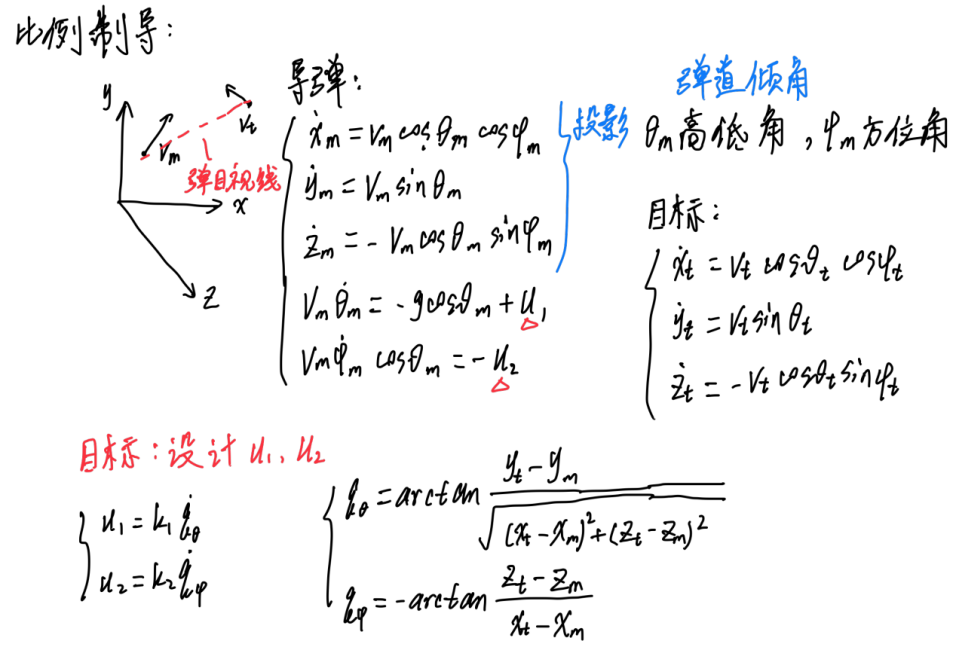


## 导弹制导率仿真（比例导引+升力限幅）

****

****

**思路：**

****

close all,clear,clc,

tic,

global m g rho S CQ vt

m = 500;

g = 9.8;

rho = 1.225;

S = 0.01;

CQ = 0.1;

vt = -72\*1000/(60\*60);

xm\_initial = 0;

ym\_initial = 5000;

vm\_initial = 500;

thetam\_initial = deg2rad(30);

xm = xm\_initial;

ym = ym\_initial;

vm = vm\_initial;

thetam = thetam\_initial;

xt\_initial = 6000;%目标纵程初值

xt = xt\_initial;

yt = 0;%地面目标

ytout = yt;

state = [xm; ym; vm; thetam; xt];

stateout = state;

q = -atan(ym/(xt-xm));

r = sqrt((xt-xm)^2+ym^2);

qdot = vm/r\*sin(q-thetam)-vt/r\*sin(q);

qout = q;

qdotout = qdot;

rout = r;

uout = [];

t = 0;

dt = 0.0005;

tout = t;

k = 10;%比例制导系数

while ym > 0

u = k\*m\*vm\*qdot + m\*g\*cos(thetam); %作为比例制导控制输入的导弹气动升力

if abs(u/m) > 20\*g %导弹过载限制

u = sign(u)\*20\*m\*g;

end

uout = [uout,u];

ke1 = stateequation(t,state,u);

ke2 = stateequation(t+0.5\*dt,state+0.5\*ke1\*dt,u);

ke3 = stateequation(t+0.5\*dt,state+0.5\*ke2\*dt,u);

ke4 = stateequation(t+dt,state+ke3\*dt,u);

state = state+1/6\*(ke1+2\*ke2+2\*ke3+ke4)\*dt;

stateout = [stateout,state];

xm = state(1);

ym = state(2);

vm = state(3);

thetam = state(4);

xt = state(5);

q = -atan(ym/(xt - xm));

r = sqrt((xt - xm)^2 + ym^2);

qdot = vm/r\*sin(q - thetam) - vt/r\*sin(q);

qout = [qout,q];

rout = [rout,r];

qdotout = [qdotout,qdot];

t = t + dt;

tout = [tout,t];

yt = 0;

ytout = [ytout,yt];

end

xmout = stateout(1,:);

ymout = stateout(2,:);

vmout = stateout(3,:);

thetamout = stateout(4,:);

xtout = stateout(5,:);

uout = [uout,uout(end)];

figure,

subplot(3,1,1),

plot(tout,vmout,'linewidth',2),

ylabel('vm(m/s)'),

grid on,

subplot(3,1,2),

plot(tout,thetamout\*180/pi,'linewidth',2),

ylabel('themam(deg)'),

grid on,

subplot(3,1,3),

plot(tout,uout,'linewidth',2),

ylabel('u(N)'),

grid on,

xlabel('t(s)'),

figure,

subplot(3,1,1),

plot(tout,qout\*180/pi,'linewidth',2),

ylabel('q(deg)'),

grid on,

subplot(3,1,2),

plot(tout,qdotout\*180/pi,'linewidth',2),

ylabel('qdot(deg/s)'),

grid on,

subplot(3,1,3),

plot(tout,rout,'linewidth',2),

ylabel('r(m)'),

xlabel('t(s)'),

grid on,

if r <= 1

disp('脱靶量(m)'),

r,

disp('脱靶量小于误差范围，导弹命中目标，制导飞行试验圆满成功!'),

end

figure,

plot(xmout,ymout,'r','linewidth',2),

hold on,

plot(xtout,ytout,'b','linewidth',2)

hold on,

plot(xm,ym,'\*r','markersize',20),

set(gca,'fontname','microsoft yahei'),

legend('导弹弹道','目标轨迹'),

grid on,

toc,

function statedot=stateequation(t,state,u)

global m g rho S CQ vt

xm = state(1);

ym = state(2);

vm = state(3);

thetam = state(4);

xt = state(5);

xmdot = vm\*cos(thetam);

ymdot = vm\*sin(thetam);

Q = 1/2\*rho\*vm^2\*S\*CQ;

vmdot = -g\*sin(thetam)-Q/m;

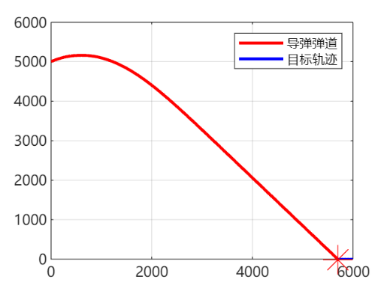
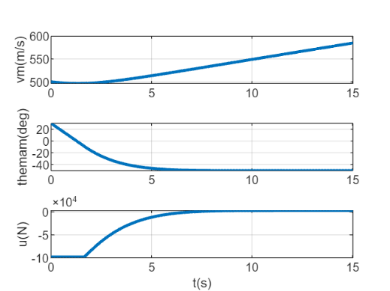
Y = u; %导弹气动升力，在此就是制导控制输入量

thetamdot = -g/vm\*cos(thetam)+Y/(m\*vm);

xtdot = vt;

statedot=[xmdot; ymdot; vmdot; thetamdot; xtdot];

end



# 模板

## 非线性系统与数值仿真基础

### ode45 函数仿真关键代码

**核心步骤：参数初始化→状态方程定义→ode45 求解→输出计算→绘图**

% 1. 参数初始化

tspan = [起始时间, 结束时间]; % 如[0,10]（第1题）、[0,20]（第2题）

x0 = [状态1初始值; 状态2初始值; ...]; % 如[5;-6]（第1题）、全0（第2题）

% 2. 调用ode45求解状态方程[t, x] = ode45(@sys\_fun, tspan, x0);

% 3. 计算输出量（按题目给定y与x关系）

y1 = 输出1表达式; % 如x(:,1)+2\*x(:,2)（第1题）、-x(:,2)+3\*x(:,1)（第2题）

y2 = 输出2表达式; % 如3\*x(:,1)-x(:,2)（第1题，单输出可省略）

% 4. 状态方程函数定义（必选，与状态维度匹配）

function dx = sys\_fun(t, x)

dx = zeros(状态维度, 1); % 如2维（第1题）、4维（第2题）

% 按题目动力学方程赋值

dx(1) = 状态1导数; % 如-x(1)+x(2)（第1题）

dx(2) = 状态2导数; % 如-x(1)-3\*x(2)-x(2)^3+5\*sin(t)（第1题）

dx(3) = 状态3导数; % 高阶系统补充（第2题）

dx(4) = 状态4导数; % 高阶系统补充（第2题）

end

### 欧拉折线法仿真关键代码

**核心步骤：步长设置→数组初始化→迭代计算→输出更新**

% 1. 基础参数初始化

tspan = [0,10]; % 同ode45仿真时间

x0 = [5;-6];

t\_step = 0.0001; % 时间步长（需小步长保证精度）

tlabel = tspan(1):t\_step:tspan(2);

tlen = length(tlabel);

% 2. 状态与输出数组初始化

x1 = zeros(tlen, 1); x2 = zeros(tlen, 1);

y1 = zeros(tlen, 1); y2 = zeros(tlen, 1);x1(1) = x0(1); x2(1) = x0(2); % 初始状态赋值y1(1) = x1(1)+2\*x2(1); y2(1) = 3\*x1(1)-x2(1); % 初始输出

% 3. 欧拉迭代计算（核心公式：x(t+Δt)=x(t)+f(t,x)\*Δt）for t = 1:tlen-1

% 计算当前状态导数

dx1 = -x1(t)+x2(t); % 同ode45状态1导数

dx2 = -x1(t)-3\*x2(t)-(x2(t))^3+5\*sin(tlabel(t)); % 同ode45状态2导数

% 更新下一时刻状态

x1(t+1) = x1(t) + dx1 \* t\_step;

x2(t+1) = x2(t) + dx2 \* t\_step;

% 更新下一时刻输出

y1(t+1) = x1(t+1)+2\*x2(t+1);

y2(t+1) = 3\*x1(t+1)-x2(t+1);

end

## PID/PD控制设计模板

### PID控制仿真模板（以质点受扰运动为例）

% 1. 全局参数与PID参数初始化

close all, clear, clc;

global m J1 J2 J3 f1 f2 f3; % 质量/转动惯量/干扰（根据题目选择）

m = 1; J1=1000; J2=1500; J3=1800; % 质点质量（2.1题）、航天器惯量（2.2题）

kp = 1; kd = 0.5; ki = 0.1; % 质点PID参数（2.1题）% kp1=-1000; kd1=-1000; ki1=-200; % 航天器PID参数（2.2题，三通道需3组）

% 2. 状态与积分项初始化

state = [位置; 速度; 姿态角1; 姿态角2; 姿态角3; 角速度1; 角速度2; 角速度3]; % 按题目维度

eta1 = 0; eta2 = 0; eta3 = 0; % 积分项（η=∫θdt，三通道对应3个）

% 3. 迭代计算（龙格-库塔4阶，dt为步长）

for t = t0:dt:tf

% 提取当前状态

x = state(1); v = state(2); theta1=state(3); omega1=state(6); % 示例

% 计算PID控制律

u = -kp\*x - kd\*v - ki\*eta1; % 质点控制律（2.1题）

% M1 = kp1\*theta1 + kd1\*omega1 + ki1\*eta1; % 航天器控制力矩（2.2题，三通道3个）

% 龙格-库塔更新状态

ke1 = statequation(t, state, u); % 状态方程函数（需自定义）

ke2 = statequation(t+0.5\*dt, state+0.5\*ke1\*dt, u);

ke3 = statequation(t+0.5\*dt, state+0.5\*ke2\*dt, u);

ke4 = statequation(t+dt, state+ke3\*dt, u);

state = state + 1/6\*(ke1 + 2\*ke2 + 2\*ke3 + ke4)\*dt;

% 更新积分项

eta1 = eta1 + theta1 \* dt; % 按积分定义更新

% 4. 状态方程函数定义（按题目动力学方程）

function output = statequation(t, state, u/M)

global m J1 J2 J3 f1 f2 f3;

% 提取状态量

x=state(1); v=state(2); theta1=state(3); omega1=state(6);

% 计算导数

xdot = v; % 质点位置导数（2.1题）

vdot = 1/m\*(u + f); % 质点速度导数（2.1题）

theta1dot = omega1 - omega2\*cos(theta1)\*tan(theta3) + omega3\*sin(theta1)\*tan(theta3); % 航天器姿态导数（2.2题）

omega1dot = (J2-J3)/J1\*omega2\*omega3 + (M1+f1)/J1; % 航天器角速度导数（2.2题）

output = [xdot; vdot; theta1dot; ...]; % 输出导数数组

end

### PD 控制关键代码（航天器姿态跟踪）

**核心差异：无积分项，控制律仅含比例 + 微分，其余同 PID**

% 1. PD参数设置（无积分项ki）

kp1 = -1000; kd1 = -1000; % 滚转通道

kp2 = -1500; kd2 = -1500; % 偏航通道

kp3 = -1800; kd3 = -1800; % 俯仰通道

% 2. PD控制律计算（无积分项）

M1 = kp1\*theta1 + kd1\*omega1; % 滚转力矩（无eta1项）

M2 = kp2\*theta2 + kd2\*omega2; % 偏航力矩

M3 = kp3\*theta3 + kd3\*omega3; % 俯仰力矩

% 3. 后续状态更新、龙格-库塔迭代、状态方程函数均与PID一致（省略重复部分）

## 倒立摆系统控制（双通道 PD + 状态反馈）

### 双通道 PD 控制关键代码（含力矩电机）

% 1. 系统参数初始化

global m1 m2 l g;

m1=1; m2=1; l=1; g=9.8; % 小车/小球质量、杆长、重力加速度

kp1=10; kd1=10; % 小车位置PD参数

kp2=50; kd2=50; % 摆角PD参数

% 2. 状态初始化

state = [x; theta; v; omega]; % x=1, theta=35°v=0.2, omega=2°/s角度转弧度

% 3. 迭代计算（龙格-库塔）

for t = t0:dt:tf

% 提取状态

x=state(1); theta=state(2); v=state(3); omega=state(4);

% 计算双通道PD控制律

u = -kp1\*x - kd1\*v; % 小车控制力

M = -kp2\*theta - kd2\*omega; % 摆角控制力矩

% 龙格-库塔更新状态（同PID步骤，调用stateequation函数）

ke1 = stateequation(t, state, u, M);

ke2 = stateequation(t+0.5\*dt, state+0.5\*ke1\*dt, u, M);

ke3 = stateequation(t+0.5\*dt, state+0.5\*ke2\*dt, u, M);

ke4 = stateequation(t+dt, state+ke3\*dt, u, M);

state = state + 1/6\*(ke1 + 2\*ke2 + 2\*ke3 + ke4)\*dt;

end

% 4. 状态方程函数（倒立摆动力学）

function statedot = stateequation(t, state, u, M)

global m1 m2 l g;

x=state(1); theta=state(2); v=state(3); omega=state(4);

% 计算导数（按题目给定动力学方程）

xdot = v;

thetadot = omega;

vdot = (m2\*g\*cos(theta)\*sin(theta) - m2\*l\*omega^2\*sin(theta) + u + cos(theta)\*M/l)/(m1 + m2\*sin(theta)^2);

omegadot = ((m1+m2)\*g\*sin(theta) - m2\*l\*omega^2\*cos(theta)\*sin(theta) + u\*cos(theta)/l + (m1+m2)\*M/(m2\*l^2))/(m1 + m2\*sin(theta)^2);

statedot = [xdot; thetadot; vdot; omegadot];

end

### 状态反馈控制关键代码（无电机，place 函数配置极点）

% 1. 系统参数与状态初始化（同3.1）

global m1 m2 l g;

m1=1; m2=1; l=1; g=9.8;

z = [x; theta; v; omega]; % 状态向量（同state）

% 2. 状态反馈增益矩阵计算（核心：place函数配置极点）

A = [0 0 1 0; 0 0 0 1; 0 m2\*g/m1 0 0; 0 (m1+m2)\*g/(m1\*l) 0 0]; % 系统矩阵

B = [0; 0; 1/m1; 1/(m1\*l)]; % 输入矩阵

lambda = [-1+i; -1-i; -2+i; -2-i]; % 期望极点（需负实部保证稳定）

K = place(A, B, lambda); % 计算反馈增益K

% 3. 迭代计算（控制律为u=-K\*z）

for t = t0:dt:tf

u = -K\*z; % 状态反馈控制律（无M，仅u）

% 龙格-库塔更新状态（同3.1，调用简化stateequation函数）

ke1 = stateequation(t, z, u);

ke2 = stateequation(t+0.5\*dt, z+0.5\*ke1\*dt, u);

ke3 = stateequation(t+0.5\*dt, z+0.5\*ke2\*dt, u);

ke4 = stateequation(t+dt, z+ke3\*dt, u);

z = z + 1/6\*(ke1 + 2\*ke2 + 2\*ke3 + ke4)\*dt;

end

% 4. 简化状态方程函数（无M，动力学方程调整）

function zdot = stateequation(t, z, u)

global m1 m2 l g;

x=z(1); theta=z(2); v=z(3); omega=z(4);

A = [m1+m2, -m2\*l\*cos(theta); -m2\*l\*cos(theta), m2\*l^2];

B = [-m2\*l\*omega^2\*sin(theta) + u; m2\*l\*g\*sin(theta)];

dotdot = inv(A)\*B; % 求解二阶导数

xdot = v;

thetadot = omega;

vdot = dotdot(1);

omegadot = dotdot(2);

zdot = [xdot; thetadot; vdot; omegadot];

end

## 复杂干扰与运动轨迹（复合干扰控制 + 火箭飞行）

### 复合干扰（常值 + 正弦）控制关键代码（增广系统 + place）

% 1. 全局参数与干扰初始化

global m Omega;

m=1; Omega=1; % 质量、干扰频率

a=5; phi=30\*pi/180; f0=6; % 干扰参数（a:振幅, phi:初相, f0:常值）

% 2. 增广系统状态初始化（含虚拟动态，5维）

state = [z1; z2; z3; z4; z5]; % z1-z3:虚拟状态, z4=x, z5=v，初始全0

% 3. 反馈增益计算（place函数）

A = [0 1 0 0 0; 0 0 1 0 0; 0 -Omega^2 0 1 0; 0 0 0 0 1; 0 0 0 0 0]; % 增广系统矩阵

B = [0; 0; 0; 0; 1/m]; % 输入矩阵

lambda = [-0.2+i; -0.2-i; -0.5+i; -0.5-i; -0.25]; % 期望极点

K = place(A, B, lambda);

% 4. 迭代计算（含干扰计算）

for t = t0:dt:tf

f = a\*sin(Omega\*t + phi) + f0; % 实时计算复合干扰

u = -K\*state; % 状态反馈控制律

% 龙格-库塔更新状态（调用增广系统stateequation）

ke1 = stateequation(t, state, u, f);

ke2 = stateequation(t+0.5\*dt, state+0.5\*ke1\*dt, u, f);

ke3 = stateequation(t+0.5\*dt, state+0.5\*ke2\*dt, u, f);

ke4 = stateequation(t+dt, state+ke3\*dt, u, f);

state = state + 1/6\*(ke1 + 2\*ke2 + 2\*ke3 + ke4)\*dt;

end

% 5. 增广系统状态方程函数

function statedot = stateequation(t, state, u, f)

global m Omega;

z1=state(1); z2=state(2); z3=state(3); z4=state(4); z5=state(5);

% 增广系统导数（按题目给定）

z1dot = z2;

z2dot = z3;

z3dot = -Omega^2\*z2 + z4;

z4dot = z5;

z5dot = 1/m\*(u + f);

statedot = [z1dot; z2dot; z3dot; z4dot; z5dot];

end

### 火箭增程弹飞行仿真关键代码（while 循环 + 分段推力）

% 1. 飞行参数初始化

m=100; g=9.8; rho=1.225; S=0.01; C\_Q=1; C\_Y=0.2; % 质量、气动参数

alpha=deg2rad(3); P=0; % 攻角、初始推力

Z = [x; y; v; theta]; % 初始状态：x=0,y=0,v=50,theta=30°转弧度

dt=0.01; t=0; zero\_count=0; flag=true; % 步长、计数（落地判断）

% 2. while循环仿真（按飞行阶段调整推力）

while flag

% 推力分段：0-10秒500N，10秒后0（题目要求发动机工作10秒）

if t>=1 && t<11

P=500;

else

P=0;

end

% 龙格-库塔计算状态导数（调用飞行力学函数）

dZ1 = calculate\_flight(Z,P,m,g,alpha,rho,S,C\_Q,C\_Y);

dZ2 = calculate\_flight(Z+0.5\*dt\*dZ1,P,m,g,alpha,rho,S,C\_Q,C\_Y);

dZ3 = calculate\_flight(Z+0.5\*dt\*dZ2,P,m,g,alpha,rho,S,C\_Q,C\_Y);

dZ4 = calculate\_flight(Z+dt\*dZ3,P,m,g,alpha,rho,S,C\_Q,C\_Y);

Z = Z + (dZ1+2\*dZ2+2\*dZ3+dZ4)\*dt/6;

% 落地判断：y<1e-1且计数>1时停止

if Z(2)<1e-1

zero\_count=zero\_count+1;

end

if zero\_count>1

flag=false;

end

t = t+dt;

end

% 3. 飞行力学函数（计算状态导数）

function dZ = calculate\_flight(Z,P,m,g,alpha,rho,S,C\_Q,C\_Y)

x=Z(1); y=Z(2); v=Z(3); theta=Z(4);

% 计算气动阻力Q、升力Y

[Q,Y] = calculate\_qidong(v,rho,S,C\_Q,C\_Y);

% 状态导数（按飞行方程）

dx = v\*cos(theta);

dy = v\*sin(theta);

dv = -g\*sin(theta) - Q/m + P\*cos(alpha);

dtheta = (-g\*cos(theta) + Y/m + P\*sin(alpha)/m)/v; % 除以速度避免零误差

dZ = [dx, dy, dv, dtheta];

end

% 4. 气动参数计算函数

function [Q,Y] = calculate\_qidong(v,rho,S,C\_Q,C\_Y)

Q = 0.5\*rho\*v^2\*S\*C\_Q; % 阻力公式

Y = 0.5\*rho\*v^2\*S\*C\_Y; % 升力公式

end

## 耦合系统与制导律（滑块摆锤 + 导弹制导）

### 滑块 - 摆锤耦合系统非线性控制关键代码

% 1. 系统参数与非线性控制律参数

global m1 m2 l g;

m1=20; m2=10; l=5; g=9.8; % 滑块/摆锤质量、杆长

kp=0.1; kd=1; % 控制律比例、微分系数

% 2. 状态初始化

state = [x; theta; v; omega]; % x=10, theta=30°转弧度, v=0, omega=0

% 3. 迭代计算（非线性控制律）for t = t0:dt:tf

% 提取状态

x=state(1); theta=state(2); v=state(3); omega=state(4);

% 非线性控制律计算（按题目给定公式）

F = -m2\*(g\*cos(theta)+l\*omega^2)\*sin(theta) ...

-kp\*(m1+m2\*sin(theta)^2)\*(m1\*x - m2\*l\*sin(theta)) ...

-kd\*(m1\*v - m2\*l\*omega\*cos(theta));

% 龙格-库塔更新状态（调用耦合系统stateequation）

ke1 = stateequation(t, state, F);

ke2 = stateequation(t+0.5\*dt, state+0.5\*ke1\*dt, F);

ke3 = stateequation(t+0.5\*dt, state+0.5\*ke2\*dt, F);

ke4 = stateequation(t+dt, state+ke3\*dt, F);

state = state + 1/6\*(ke1 + 2\*ke2 + 2\*ke3 + ke4)\*dt;

end

% 4. 耦合系统状态方程函数

function statedot = stateequation(t, state, F)

global m1 m2 l g;

x=state(1); theta=state(2); v=state(3); omega=state(4);

% 导数计算（按耦合动力学方程）

xdot = v;

thetadot = omega;

vdot = (F + m2\*l\*omega^2\*sin(theta) + m2\*g\*sin(theta)\*cos(theta))/(m1 + m2\*sin(theta)^2);

omegadot = -(F\*cos(theta) + m2\*l\*omega^2\*sin(theta)\*cos(theta) + (m1+m2)\*g\*sin(theta))/((m1 + m2\*sin(theta)^2)\*l);

statedot = [xdot; thetadot; vdot; omegadot];

end

### 导弹制导律仿真关键代码（比例导引 + 升力限幅）

% 1. 导弹与目标参数初始化

m=500; g=9.8; rho=1.225; S=0.01; C\_Q=0.1; % 导弹质量、气动参数

vt=-20; xt=6000; % 目标速度（-72km/h转m/s）、初始位置

X = [xm; ym; vm; thetam]; % 导弹初始状态：xm=0,ym=5000,vm=500,thetam=0°

k=10; dt=0.01; t=0; % 制导律系数、步长

% 2. 制导仿真循环（至命中或落地）[r,q,dq] = calculate\_rqdq(X,xt,vt);

% 初始弹目距离、视线角、视线角速率

while r>0 && X(2)>0 % 条件：弹目距离>0且高度>0

% 计算气动升力Y（带限幅：|Y|≤20mg）

[Q,Y] = calculate\_QY(dq,X(3),X(4),rho,k,m,g,S,C\_Q);

% 龙格-库塔更新导弹状态

dX1 = calculate\_dX(t,X,Q,Y,m,g);

dX2 = calculate\_dX(t,X+0.5\*dt\*dX1,Q,Y,m,g);

dX3 = calculate\_dX(t,X+0.5\*dt\*dX2,Q,Y,m,g);

dX4 = calculate\_dX(t,X+dt\*dX3,Q,Y,m,g);

X = X + (dX1+2\*dX2+2\*dX3+dX4)\*dt/6;

% 更新目标位置、弹目参数

xt = xt + vt\*dt;

[r,q,dq] = calculate\_rqdq(X,xt,vt);

t = t+dt;

end

% 3. 导弹状态导数计算函数

function dX = calculate\_dX(t,X,Q,Y,m,g)

xm=X(1); ym=X(2); vm=X(3); thetam=X(4);

dxm = vm\*cos(thetam);

dym = vm\*sin(thetam);

dvm = -g\*sin(thetam) - Q/m;

dthetam = (-g\*cos(thetam) + Y/m)/max(vm,1e-3); % 避免vm=0

dX = [dxm,dym,dvm,dthetam];

end

% 4. 升力计算与限幅函数

function [Q,Y] = calculate\_QY(dq,vm,thetam,rho,k,m,g,S,C\_Q)

Q = 0.5\*rho\*vm^2\*S\*C\_Q; % 阻力

Y\_temp = k\*m\*vm\*dq + m\*g\*cos(thetam); % 比例导引升力

% 限幅：|Y|≤20mg

if Y\_temp>20\*m\*g

Y=20\*m\*g;

elseif Y\_temp<-20\*m\*g

Y=-20\*m\*g;

else

Y=Y\_temp;

end

end

% 5. 弹目参数（距离、视线角、视线角速率）计算函数

function [r,q,dq] = calculate\_rqdq(X,xt,vt)

xm=X(1); ym=X(2); vm=X(3); thetam=X(4);

r = sqrt((xm-xt)^2 + ym^2); % 弹目距离

q = -atan(ym/(xt-xm)); % 视线角

dq = (vm\*sin(q-thetam) - vt\*sin(q))/r; % 视线角速率

end

## 通用

**绘图模块：所有题型绘图逻辑一致，核心代码如下（按需调用）**

% 单图多曲线

figure;

hold on; grid on;

plot(t, y1, 'r', 'LineWidth',2);

plot(t, y2, 'b', 'LineWidth',2);

xlabel('t(s)'); ylabel('输出量');

legend('y1','y2');

% 多子图（以2行1列为例）

figure;

subplot(2,1,1);

hold on; grid on;

plot(t, y1, 'b', 'LineWidth',2);

xlabel('t(s)'); ylabel('y1');

subplot(2,1,2);

hold on; grid on;

plot(t, y2, 'b', 'LineWidth',2);

xlabel('t(s)'); ylabel('y2');