目录

一、 非线性系统与数值仿真基础（第一次上机）

1.1 Ode45 函数仿真

第 1 题：

已知某非线性系统状态空间描述如下

状态方程 $\begin{cases}\dot{x}_1(t)=-x_1+x_2\\ \dot{x}_2(t)=-x_1-3x_2-x_2^3+5\sin t\end{cases}$，输出方程 $\begin{cases}y_1=x_1+2x_2\\ y_2=3x_1-x_2\end{cases}$

系统初始状态 $\begin{cases}x_1(0)=5\\ x_2(0)=-6\end{cases}$

请调用 ode45 函数对上述系统进行仿真，并绘制出函数 $y_1(t),y_2(t)$ 的曲线。具体要求如下：

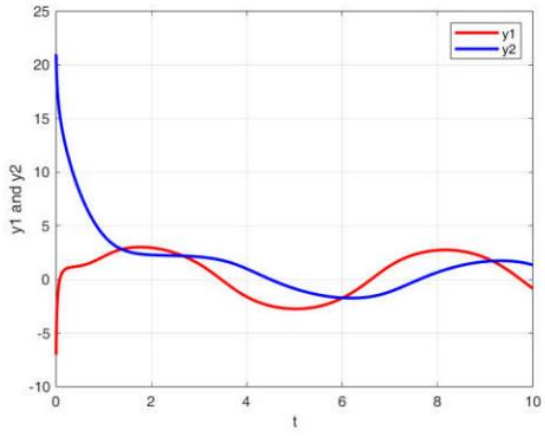
（1）仿真时间区间为[0,10]秒；

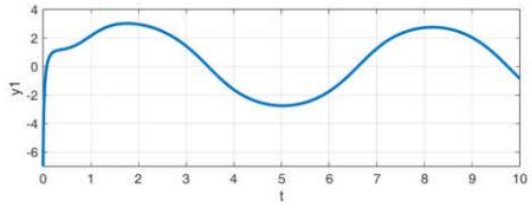
（2）第一幅仿真结果图中，两曲线绘制在一个图中，一个曲线用红色('r')，另一个曲线用蓝色('b')，并用图例 legend 函数将其区分开来；

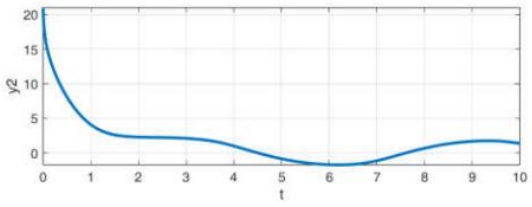
（3）第二幅仿真结果图中，包括上下两个子图，分别绘制这两条曲线；

（4）所有的图，曲线粗细设置为 2 个单位（'linewidth'），坐标轴加标注(xlabel, ylabel)，图形加网格(grid on)

期望的绘制效果如下：

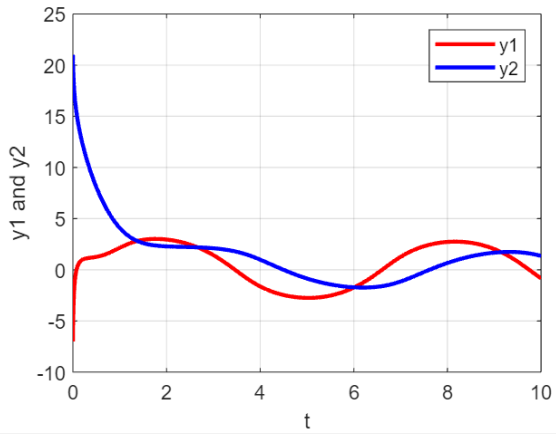


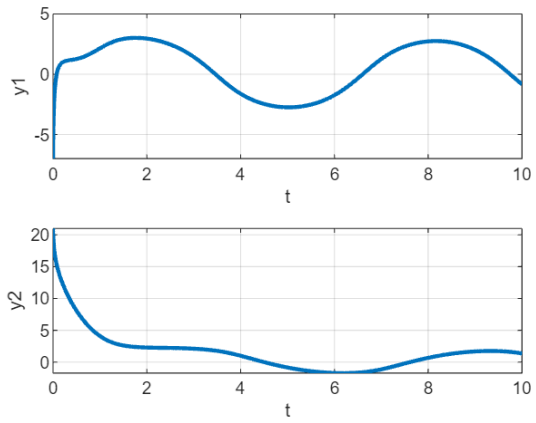




```
%% 初始设置
close all,clear,clc,
tic,
% 初始化仿真时间区间
t0 = 0.0;
tf = 10.0;

%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
```





1.2 Ode45 函数仿真

第 2 题：

对于如下系统

$$y^{(4)}+6y^{(3)}+18\ddot{y}+24\dot{y}+16y=-\dot{u}+3u$$

输入信号为

$$u(t)=5e^{-0.5t}\cos(t)+6\sin(t+\pi/3)$$

系统初始条件为 0,即 $y(0)=\dot{y}(0)=\ddot{y}(0)=\dddot{y}(0)=0$

试调用 ode45 函数对系统进行仿真，并绘制仿真结果。

要求：

（1）仿真时间设定为 20 秒钟；

（2）调用绘图函数 plot 绘制曲线 $y(t)$，给坐标轴加标注，图形加网格；

（3）线条粗细设定为 2 个单位。

思路推导：

方程形式变换：

假设 $z^{(4)}+6\dddot{z}+18\ddot{z}+24\dot{z}+16z=u$

即 $\left(\frac{d^4}{dt^4}+6\frac{d^3}{dt^3}+18\frac{d^2}{dt^2}+24\frac{d}{dt}+16\right)z=u$，两边同乘 $\left(2\frac{d}{dt}+3\right)$

得 $\left(\frac{d^4}{dt^4}+6\frac{d^3}{dt^3}+18\frac{d}{dt^2}+24\frac{d}{dt}+16\right)\left(2\frac{d}{dt}+3\right)z=\left(2\frac{d}{dt}+3\right)u$

和原式对比得 $y=2\dot{z}+3z$ ⇒ 要输出的 yout

求初值：

$$\begin{cases}3z(0)+2\dot{z}(0)=y(0)=1\\3\dot{z}(0)+2\ddot{z}(0)=\dot{y}(0)=2\\3\ddot{z}(0)+2\dddot{z}(0)=\ddot{y}(0)=-3\\3\dddot{z}(0)+2z^{(4)}(0)=\dddot{y}(0)=-2\\16z(0)+24\dot{z}(0)+18\ddot{z}(0)+6\dddot{z}(0)+z^{(4)}(0)=u(0)\end{cases}$$

令 $z=[z(0),\ \cdots,\ z^{(4)}(0)]^T$ 解方程 $A\cdot z=B$ ⇒ $z=A^{-1}\cdot B$

$$A=\begin{bmatrix}3&2&0&0&0\\0&3&2&0&0\\0&0&3&2&0\\0&0&0&3&2\\16&24&18&6&1\end{bmatrix}\quad B=\begin{bmatrix}1\\2\\-3\\-2\\u(0)\end{bmatrix}$$

微分方程求解：

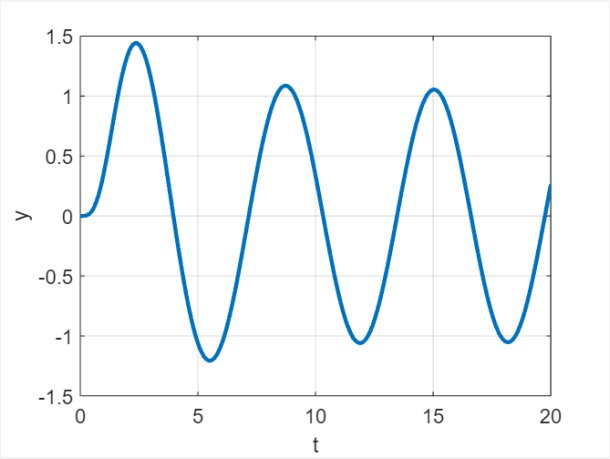
令 $\begin{cases} x_1 \triangleq z \\ x_2 \triangleq \dot{z} \\ x_3 \triangleq \ddot{z} \\ x_4 \triangleq \dddot{z} \end{cases}$

列状态方程：$\Rightarrow$ matlab中用ode45求解

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = x_3 \\ \dot{x}_3 = x_4 \\ \dot{x}_4 = -6x_4 - 18x_3 - 24x_2 - 16x_1 + u \end{cases}$$

```
%% 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
```



1.3 欧拉折线法仿真 1.1

第 3 题（探索性选做题目）；

对于上述第 1 题，请自行探索采用欧拉折线法的方法进行系统仿真求解，而非调用人家的 ode45 函数。

已知，对于如下动态系统

$$\begin{cases}\dot{x} = f(t,x), \ x(0) = x_0\end{cases}$$

欧拉折线法数值求解公式为

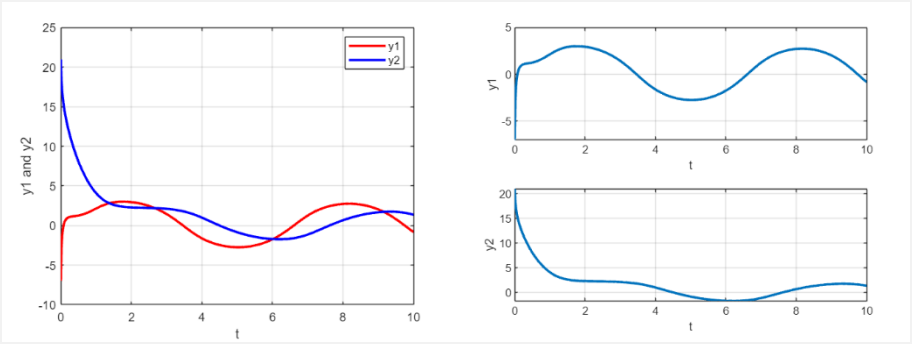
$$x(t+\Delta t) = x(t) + f(t,x)\cdot\Delta t$$

仿真结果图的绘制如第 1 题。

```
%% 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
```



二、 PID/PD 控制

2.1 PID 控制

第 1 题：

质点单自由度受扰运动方程为 $\ddot{x}=\frac{1}{m}(u+f)$，其中 x 是质点位置坐标，u 是控制量，f 是外部干扰力，并且假设是常值干扰，但其具体数值不知道。试采用如下 PID 控制

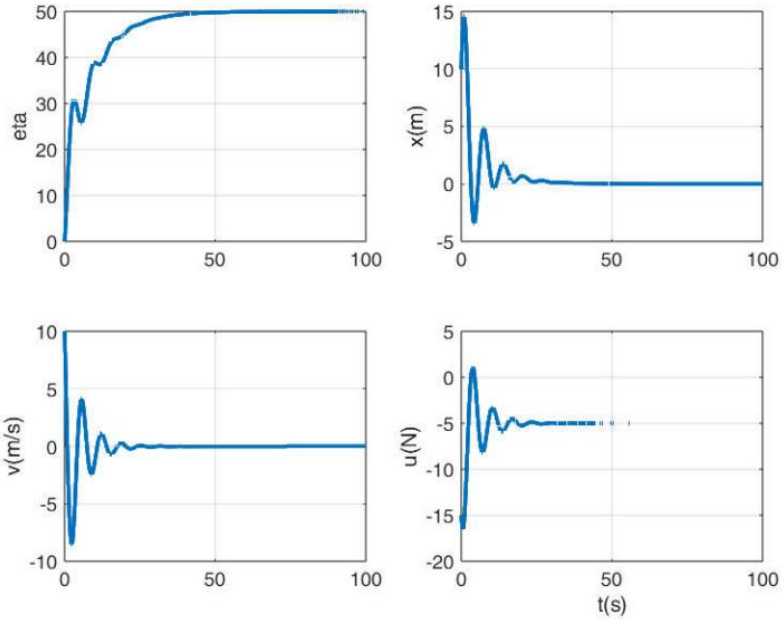
$$u=-k_{\mathrm{p}}x-k_{\mathrm{d}}\dot{x}-k_{\mathrm{i}}\int_0^t x\mathrm{d}t$$

将质点位置保持在零。

已知质点质量 $m=1(\mathrm{kg})$，控制器参数如下

$$\begin{cases}k_{\mathrm{p}}=1,\ k_{\mathrm{d}}=0.5,\ k_{\mathrm{i}}=0.1\end{cases}$$

仿真效果图如下：



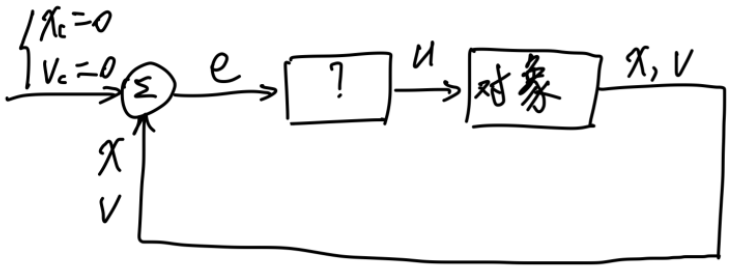
图中左上子图表示位置坐标对时间的积分变量 $\eta(t)$ 的曲线。

PD 控制和 PID 控制思路：

PD控制：常值干扰

$m\ddot{x} = u + f$ ，$u = u(x, \dot{x})$ $\begin{cases} x \to 0 \\ \dot{x} \to 0 \end{cases}$ $V = \dot{x}$

$u = -k_p(x - x_c) - k_d(V - V_c) = -k_p x - k_d V = -k_p x - k_d \dot{x}$



改用PID控制：

$u = -k_p(x - x_c) - k_d(V - V_c) - k_I \int_0^t (x - x_c)\,dt$

ξ

⇓逆耀公式

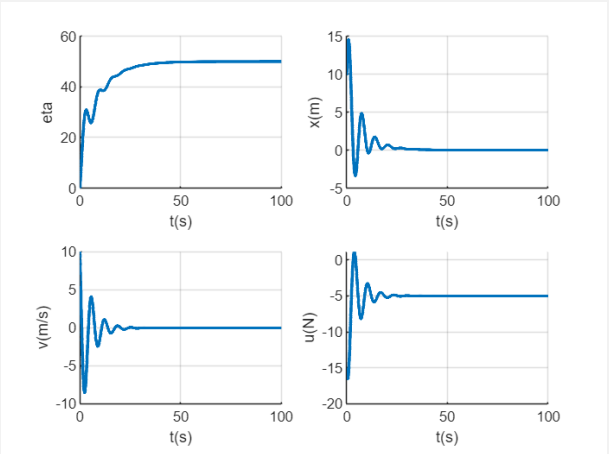
$f(t+\Delta t) = f(t) + \dot{f}(t) \cdot \Delta t$

法二：拓展为增广：$\dot{\xi} = x$ → 一起放入状态方程中

```
%% 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
```



2.2 PD 控制

第 2 题：

已知航天器姿态运动方程如下

$$\begin{cases}\dot{\theta}_1=\omega_1-\omega_2\cos\theta_1\tan\theta_3+\omega_3\sin\theta_1\tan\theta_3\\ \dot{\theta}_2=\omega_2\dfrac{\cos\theta_1}{\cos\theta_3}-\omega_3\dfrac{\sin\theta_1}{\cos\theta_3}\\ \dot{\theta}_3=\omega_2\sin\theta_1+\omega_3\cos\theta_1\end{cases}$$

$$\begin{cases}\dot{\omega}_1=\dfrac{J_2-J_3}{J_1}\omega_2\omega_3+\dfrac{1}{J_1}(M_1+f_1)\\ \dot{\omega}_2=\dfrac{J_3-J_1}{J_2}\omega_3\omega_1+\dfrac{1}{J_2}(M_2+f_2)\\ \dot{\omega}_3=\dfrac{J_1-J_2}{J_3}\omega_1\omega_2+\dfrac{1}{J_3}(M_3+f_3)\end{cases}$$

其中 J_1,J_2,J_3 分别表示三轴转动惯量，$\theta_1,\theta_2,\theta_3$ 分别表示滚转角、偏航角、俯仰角，$\omega_1,\omega_2,\omega_3$ 分别表示滚转角速率、偏航角速率、俯仰角速率，M_1,M_2,M_3 分别表示滚转控制力矩、偏航控制力矩、俯仰控制力矩，f_1,f_2,f_3 分别表示三轴干扰力矩。

假设系统参数

$$\begin{cases}J_1=1000(kg.m^2)\\ J_2=1500(kg.m^2)\\ J_3=1800(kg.m^2)\end{cases}$$

系统初始状态

$$\begin{cases}\theta_{10}=10^o\\ \theta_{10}=20^o\\ \theta_{30}=-30^o\end{cases},\quad\begin{cases}\omega_{10}=10^o/s\\ \omega_{10}=20^o/s\\ \omega_{30}=-30^o/s\end{cases}$$

三轴外部干扰力矩

$$\begin{cases}f_1=200(N.m)\\ f_2=300(N.m)\\ f_3=360(N.m)\end{cases}$$

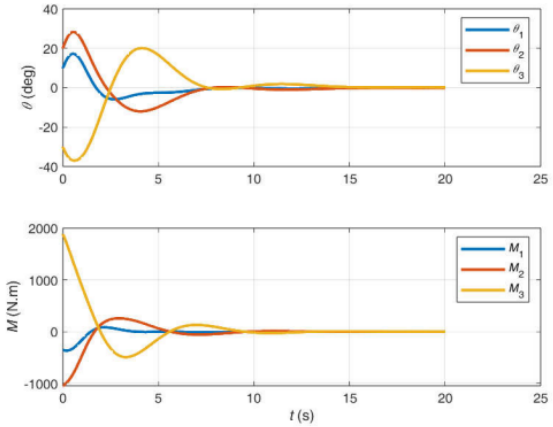
试设计三通道独立的 PID 控制器

$$\begin{cases}M_1=k_{i1}\eta_1+k_{p1}\theta_1+k_{d1}\omega_1\\ M_2=k_{i2}\eta_2+k_{p2}\theta_2+k_{d2}\omega_2\\ M_3=k_{i3}\eta_3+k_{p3}\theta_3+k_{d3}\omega_3\end{cases}$$

其中

$$\eta_i=\int_0^t\theta_i\mathrm{d}t\quad(i=1\sim3)$$

使航天器姿态角稳定在 0^o，并通过数值仿真检验所设计控制器的有效性，仿真效果如下：



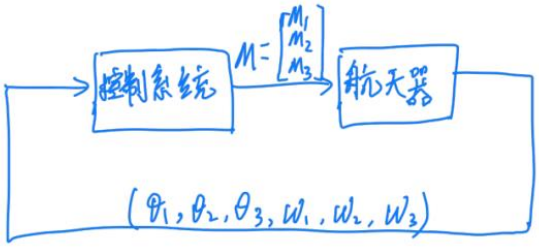
思路：

刚体三轴控制：

例4：

$$\begin{cases} \dot{\theta}_1 = w_1 - w_2\cos\theta_1\tan\theta_3 + w_3\sin\theta_1\tan\theta_3 \\ \dot{\theta}_2 = w_2\dfrac{\cos\theta_1}{\cos\theta_3} - w_3\dfrac{\sin\theta_1}{\cos\theta_3} \\ \dot{\theta}_3 = w_2\sin\theta_1 + w_3\cos\theta_1 \end{cases}$$

$$\begin{cases} \dot{w}_1 = \dfrac{J_2-J_3}{J_1}w_2w_3 + \dfrac{M_1}{J_1} \\ \dot{w}_2 = \dfrac{J_3-J_1}{J_2}w_1w_3 + \dfrac{M_2}{J_2} \\ \dot{w}_3 = \dfrac{J_1-J_2}{J_3}w_1w_2 + \dfrac{M_3}{J_3} \end{cases}$$



人为解耦

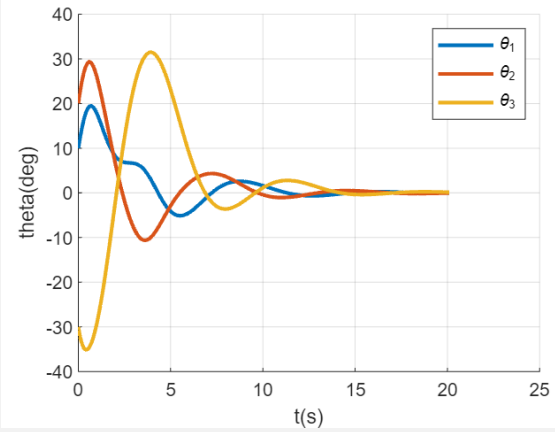
$$\begin{cases} M_1 = -k_{p_1}\theta_1 - k_{d_1}w_1 = u_1 \\ M_2 = -k_{p_2}\theta_2 - k_{d_2}w_2 = u_2 \\ M_3 = -k_{p_3}\theta_3 - k_{d_3}w_3 = u_3 \end{cases}$$

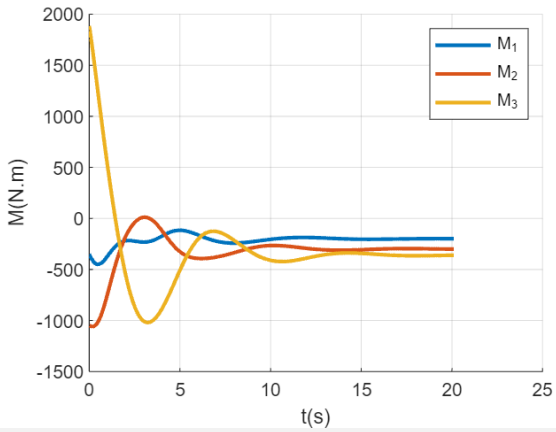
```
%% 初始参数设置
close all,clear,clc,
tic,

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
```





2.3 三通道独立 PD 控制航天器三轴姿态

第 3 题：（探索性题目，选做）

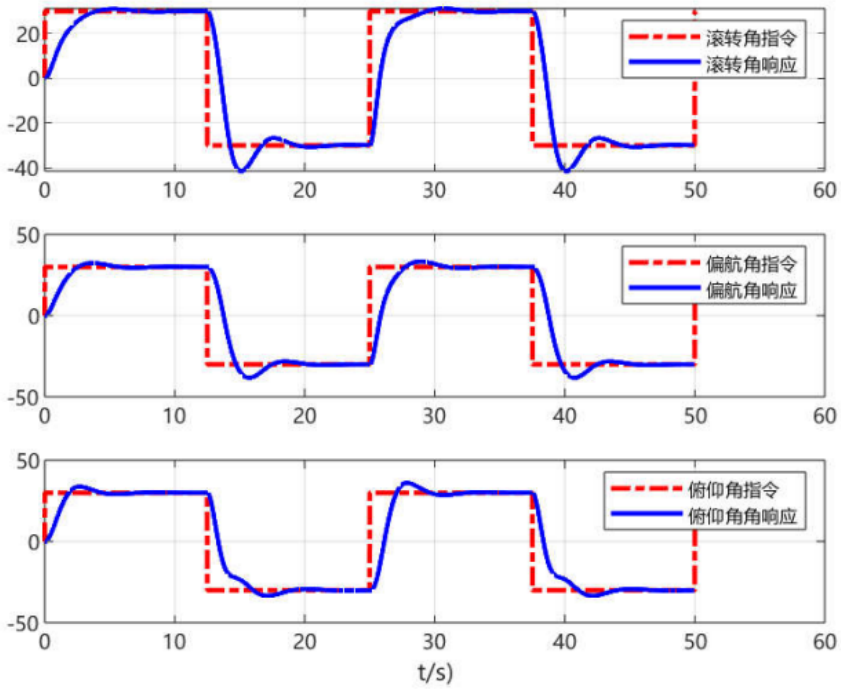
已知航天器姿态运动方程如下

$$\begin{cases} \dot{\theta}_1 = \omega_1 - \omega_2 \cos\theta_1 \tan\theta_3 + \omega_3 \sin\theta_1 \tan\theta_3 \\ \dot{\theta}_2 = \omega_2 \dfrac{\cos\theta_1}{\cos\theta_3} - \omega_3 \dfrac{\sin\theta_1}{\cos\theta_3} \\ \dot{\theta}_3 = \omega_2 \sin\theta_1 + \omega_3 \cos\theta_1 \end{cases}$$

$$\begin{cases} \dot{\omega}_1 = \dfrac{J_2 - J_3}{J_1}\omega_2\omega_3 + \dfrac{1}{J_1}M_1 \\ \dot{\omega}_2 = \dfrac{J_3 - J_1}{J_2}\omega_3\omega_1 + \dfrac{1}{J_2}M_2 \\ \dot{\omega}_3 = \dfrac{J_1 - J_2}{J_3}\omega_1\omega_2 + \dfrac{1}{J_3}M_3 \end{cases}$$

三轴转动惯量参数如第二题。

要求设计三通道独立的 PD 控制器，使航天器三轴姿态分别跟踪幅度为 30°，周期为 25 秒的方波。仿真效果图如下：

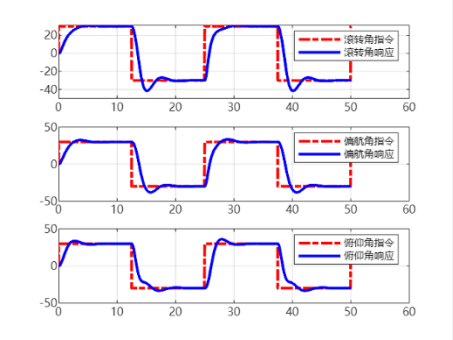



```
%% 参数设置
close all,clear,clc,
tic,
global J1 J2 J3;
J1 = 1000;  J2 = 1500;  J3 = 1800;

% 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);
```

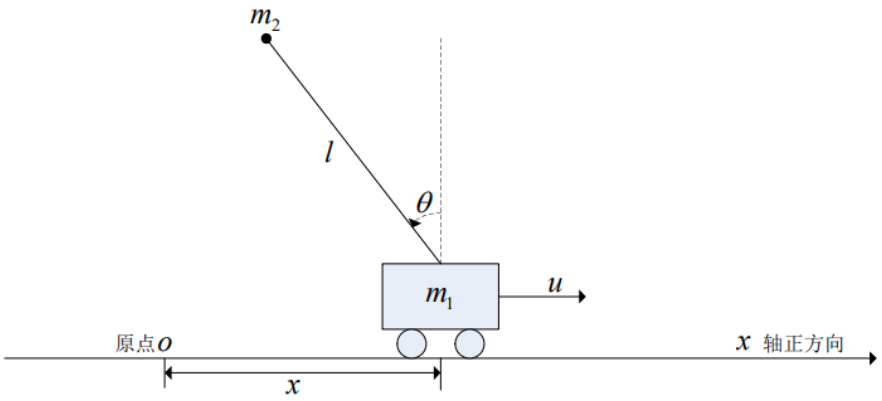
```
set(gca, 'fontname', 'microsoft yahei');
legend('偏航角指令', '偏航角响应');
grid on;

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
```



三、 倒立摆系统控制（双通道 PD+状态反馈）

3.1 双通道 PD 控制（含力矩电机）

第 1 题：



图：单级倒立摆系统

质量为m_1的小车置于光滑水平面上，质量为m_2的小球通过长度为l的轻质杆铰接于小车上（铰接处光滑无摩擦）。小车的水平位置坐标为$x(t)$如图示，轻质杆与竖直参考线的夹角为$\theta(t)$如图示（规定图示θ为正）。。

对小车施以水平控制力u，如图所示，同时在杆与小车铰接处安装一个力矩电机，其输出力矩为M（图中未画出）。

已知系统动力学方程如下

$$\begin{cases}\ddot{x}=\dfrac{m_2 g\cos\theta\sin\theta-m_2 l\dot{\theta}^2\sin\theta}{m_1+m_2\sin^2\theta}+\dfrac{1}{m_1+m_2\sin^2\theta}u+\dfrac{\cos\theta}{(m_1+m_2\sin^2\theta)l}M\\ \ddot{\theta}=\dfrac{(m_1+m_2)g\sin\theta-m_2 l\dot{\theta}^2\cos\theta\sin\theta}{(m_1+m_2\sin^2\theta)l}+\dfrac{\cos\theta}{(m_1+m_2\sin^2\theta)l}u+\dfrac{m_1+m_2}{(m_1+m_2\sin^2\theta)m_2 l^2}M\end{cases}$$

试设计两通道独立的 PD 控制器

$$\begin{cases}u=-k_{p1}x-k_{d1}\dot{x}\\ M=-k_{p2}\theta-k_{d2}\dot{\theta}\end{cases}$$

使得小车位置和摆杆摆角同时趋零，即$x\to 0,\theta\to 0$。

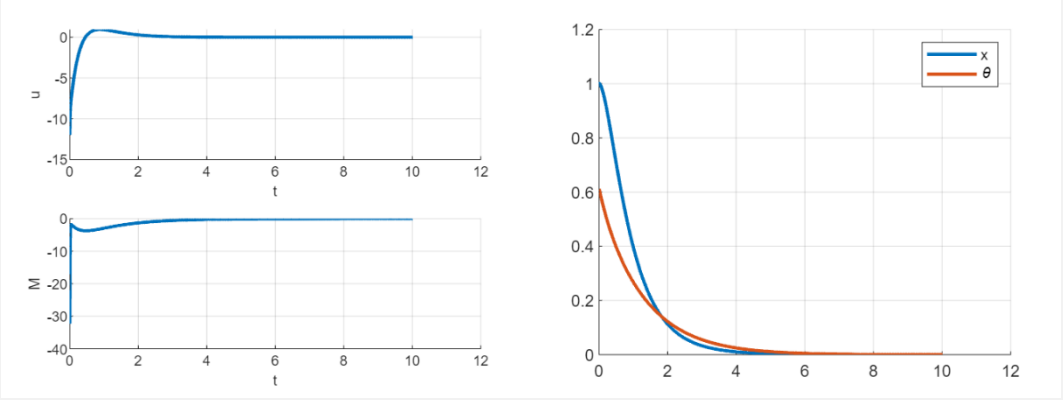
已知：$m_1=m_2=1(\mathrm{kg})$，$l=1(\mathrm{m})$，$g=9.8(\mathrm{m/s^2})$

系统初始状态：$\{x=1(\mathrm{m}),\ \theta=35^\circ,\ \dot{x}=0.2(\mathrm{m/s}),\ \dot{\theta}=2^\circ/\mathrm{s}$

```
%% 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
```



3.2 状态反馈控制（无电机，place 函数配置极点）

第 2 题：

将上述倒立摆中的执行电机拆掉，系统动力学方程如下

$$\begin{cases}\ddot{x}=\dfrac{m_2 g\cos\theta\sin\theta-m_2 l\dot{\theta}^2\sin\theta}{m_1+m_2\sin^2\theta}+\dfrac{1}{m_1+m_2\sin^2\theta}u\\ \ddot{\theta}=\dfrac{(m_1+m_2)g\sin\theta-m_2 l\dot{\theta}^2\cos\theta\sin\theta}{(m_1+m_2\sin^2\theta)l}+\dfrac{\cos\theta}{(m_1+m_2\sin^2\theta)l}u\end{cases}$$

试设计控制律

$u=-k_1x-k_2\theta-k_3\dot{x}-k_4\dot{\theta}$

使得小车位置和摆杆摆角同时趋零，即 $x\to 0,\theta\to 0$。

系统参数和初始状态仍如第 1 题。

思路：

倒立摆 PD 控制：

$$\begin{cases}\ddot{x}=\dfrac{m_2 g\cos\theta\sin\theta-m_2 l\dot{\theta}^2\sin\theta}{m_1+m_2\sin^2\theta}+\dfrac{1}{m_1+m_2\sin^2\theta}u+\dfrac{\cos\theta}{(m_1+m_2\sin^2\theta)l}M\\ \ddot{\theta}=\dfrac{(m_1+m_2)g\sin\theta-m_2 l\dot{\theta}^2\cos\theta\sin\theta}{(m_1+m_2\sin^2\theta)l}+\dfrac{\cos\theta}{(m_1+m_2\sin^2\theta)l}u+\dfrac{m_1+m_2}{(m_1+m_2\sin^2\theta)m_2 l^2}M\end{cases}$$

PD 控制：$\begin{cases}u=-k_{p_1}x-k_{d_1}\dot{x}\\ M=-k_{p_2}\theta-k_{d_2}\dot{\theta}\end{cases}$

变式：去掉杆的控制电机，去掉 M：

$$\begin{cases}\ddot{x}=\dfrac{m_2 g\cos\theta\sin\theta-m_2 l\dot{\theta}^2\sin\theta}{m_1+m_2\sin^2\theta}+\dfrac{1}{m_1+m_2\sin^2\theta}u\\ \ddot{\theta}=\dfrac{(m_1+m_2)g\sin\theta-m_2 l\dot{\theta}^2\cos\theta\sin\theta}{(m_1+m_2\sin^2\theta)l}+\dfrac{\cos\theta}{(m_1+m_2\sin^2\theta)l}u\end{cases}$$

设计 $u=-k_1x-k_2\theta-k_3\dot{x}-k_4\dot{\theta}=-kx$

思想 1：从 x 和 u 关系构造 x 和 $\dot{x}$ 状态方程（状态反馈特征值法）

$\dot{x}=Ax+Bu=(A+Bk)x$ ← 调 place 函数求 $k=\text{place}(A,B,\lambda)$

思想 2：$\theta\to 0$，$x\to 0$ 时近似，将上式线性化，忽略二阶项保留一阶项

$$\begin{cases}\dot{x}=v\\ \dot{\theta}=w\\ \dot{v}=\dfrac{m_2 g}{m_1}\theta+\dfrac{1}{m_1}u\\ \dot{w}=\dfrac{(m_1+m_2)g}{m_1 l}\theta+\dfrac{1}{m_1 l}u\end{cases}$$

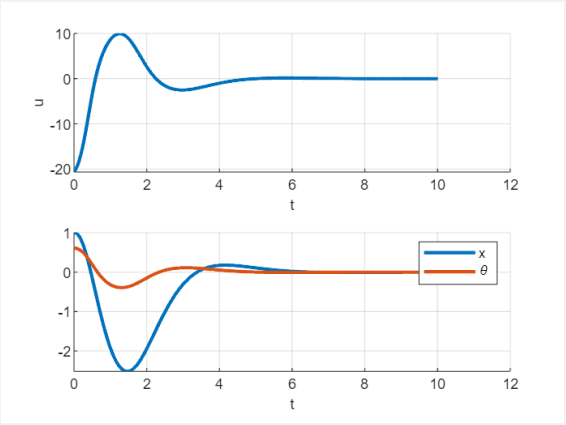
矩阵形式

$$\begin{bmatrix}\dot{x}\\ \dot{\theta}\\ \dot{v}\\ \dot{w}\end{bmatrix}=\begin{bmatrix}0&0&1&0\\ 0&0&0&1\\ 0&\frac{m_2 g}{m_1}&0&0\\ 0&\frac{(m_1+m_2)g}{m_1 l}&0&0\end{bmatrix}\begin{bmatrix}x\\ \theta\\ v\\ w\end{bmatrix}+\begin{bmatrix}0\\ 0\\ \frac{1}{m_1}\\ \frac{1}{m_1 l}\end{bmatrix}u$$

```
%% 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
```



四、 复杂干扰与运动轨迹（复合干扰控制+火箭飞行）

4.1 符合干扰（常值+正弦）控制（增广系统+place）

第 1 题：

质点单自由度受扰动力学方程是

$$\ddot{x}=\frac{1}{m}(u+f) \quad (1)$$

x 表示其位置坐标，u 是控制力，$f=f_0+a\sin(\Omega t+\phi)$ 是外部干扰，且 Ω 是已知且恒定的，f_0,a,ϕ 是未知且恒定的。试设计闭环控制律 u，使输出 x 稳定在零值。

仿真入口参数：

$$\begin{cases} m=1(\text{kg}),\ \Omega=1(\text{rad/s}) \\ a=5(\text{N}),\ \phi=30^\circ,\ f_0=6(\text{N}) \\ x(0)=0(\text{m}),\ \dot{x}(0)=0(\text{m/s}) \end{cases}$$

提示：

现在需要构造一个虚拟的动态，以产生常值加正弦信号，该动态有两个模态，常值模态和正弦振荡模态，其对应的复频域特征值就是 $s_1=0,s_{2,3}=\pm j\Omega$．这样的线性系统其特征多项式是

$$D(s)=(s-s_1)(s-s_2)(s-s_3)=s(s+j\Omega)(s-j\Omega)=s(s^2+\Omega^2)=s^3+\Omega^2 s$$

即

$$D(s)=s^3+\Omega^2 s \quad (2)$$

以此特征多项式构造传函，并以位置误差 x 驱动的线性系统复频域描述为

$$y=\frac{1}{s^3+\Omega^2 s}x \quad (3)$$

其时域高阶微分方程为

$$\dddot{y}+\Omega^2\dot{y}=x \quad (4)$$

定义状态变量

$$\{\eta_1=y,\ \eta_2=\dot{y},\ \eta_3=\ddot{y} \quad (5)$$

则其状态空间实现如下

$$\begin{cases} \dot{\eta}_1=\eta_2 \\ \dot{\eta}_2=\eta_3 \\ \dot{\eta}_3=-\Omega^2\eta_2+x \end{cases} \quad (6)$$

原受控对象写成状态方程形式如下

$$\begin{cases} \dot{x}=v \\ \dot{v}=\frac{1}{m}(u+f) \end{cases} \quad (7)$$

将式（6）（7）合并成增广系统如下

$$\begin{cases} \dot{\eta}_1=\eta_2,\ \dot{\eta}_2=\eta_3 \\ \dot{\eta}_3=-\Omega^2\eta_2+x \\ \dot{x}=v,\ \dot{v}=\frac{1}{m}(u+f) \end{cases} \quad (8)$$

定义状态变量

$$\{z_1=\eta_1,z_2=\eta_2,z_3=\eta_3,z_4=x,z_5=v \quad (9)$$

则

$$\begin{cases} \dot{z}_1 = z_2,\ \dot{z}_2 = z_3 \\ \dot{z}_3 = -\Omega^2 z_2 + z_4 \\ \dot{z}_4 = z_5,\ \dot{z}_5 = \dfrac{1}{m}(u+f) \end{cases}$$

写成矩阵形式如下

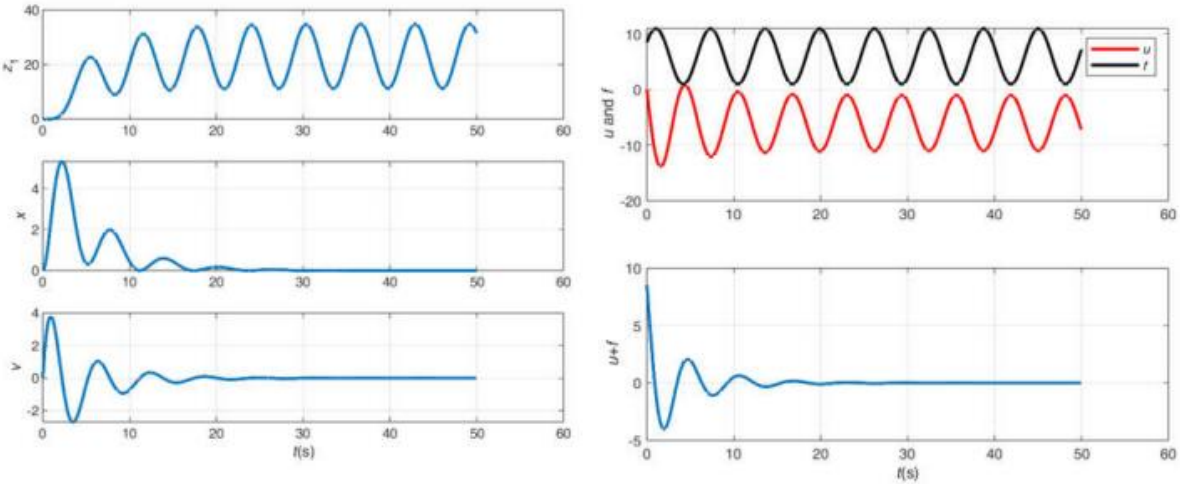
$$\dot{z} = Az + B(u+f) \quad (10)$$

其中

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & -\Omega^2 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad B = (0,0,0,0,\frac{1}{m})^{\mathrm{T}} \quad (11)$$

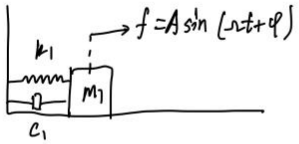
只须设计状态反馈控制律 $u = -Kz$，使得式（10）的闭环系统 $\dot{z} = (A - BK)z + Bf$ 是稳定的线性系统，即其状态矩阵 $\overline{A} = A - BK$ 的特征值均具有负实部即可。这可以容易地通过 place 函数实现，此不赘述。

可望的仿真结果如下：



思路：

外部突变干扰下的控制：



$m_1\ddot{x}_1 = -k_1x_1 - c_1\dot{x}_1 + f$ ，$\zeta_0=1$，$a=0.5$，$x_0=\zeta_0+\frac{1}{2}a=1.25$

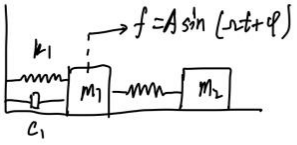
$$\Updownarrow$$

$$\begin{cases}\dot{x}_1 = v_1\\ \dot{v}_1 = -\frac{k_1}{m_1}x_1 - \frac{c_1}{m_1}v_1 + \frac{f}{m_1}\end{cases}$$

最终 m_1 作正弦来回运动

变式1：2个物块：

控制目的：m_1 最终不动，m_2 正弦振荡



$$\begin{cases}m_1\ddot{x}_1 = -k_1x_1 - c_1\dot{x}_1 + k_2(x_2-x_1) + f\\ m_2\ddot{x}_2 = -k_2(x_2-x_1)\end{cases}$$

$\because m_2\ddot{x}_2 = -k_2x_2$，$w=\sqrt{\frac{k_2}{m_2}}=\omega$，$k_2=m_2\omega^2$ — 利用共振吸收能量

$$\Rightarrow \begin{cases}\ddot{x}_1 = \cdots\\ \ddot{x}_2 = \cdots\end{cases}$$

变式2：纯控制研究（机算机模拟）基于内部原理的鲁棒法

$\ddot{x}_1 = \frac{1}{m_1}u + \frac{1}{m_1}f$，$u = -(k_1+m_2\omega^2)x_1 - c_1\dot{x}_1 + m_2\omega^2x_2$，

设 $u = -k_p x_1 - k_d\dot{x}_1 + \underbrace{k_0x_2}_{\text{虚构}}$　　$\ddot{x}_2 = -\omega^2x_2 + \omega^2x_1$

解：

$$\begin{cases}\dot{x}=v\\ \dot{v}=\frac{1}{m}(u+f)\end{cases}\qquad f=A\sin(\omega t+\varphi_0)$$

构造一个正弦波满足和 x 的关系：

$\ddot{\zeta}+\omega^2\zeta=x$，令 $z_1=\zeta$，$z_2=\dot{\zeta}$，$z_3=x$，$z_4=v$

闭环

$$\Leftrightarrow\text{微分方程形式}\quad\begin{cases}\dot{z}_1=z_2\\ \dot{z}_2=z_3-\omega^2z_1\\ \dot{z}_3=z_4\\ \dot{z}_4=\frac{1}{m}(u+f)\end{cases}$$

→ 目的：计算机中模拟+实际模型的控制器，达到和变式1同样效果

设 $\dot{z}=Az+Bu+Bf$，$u=-kz$

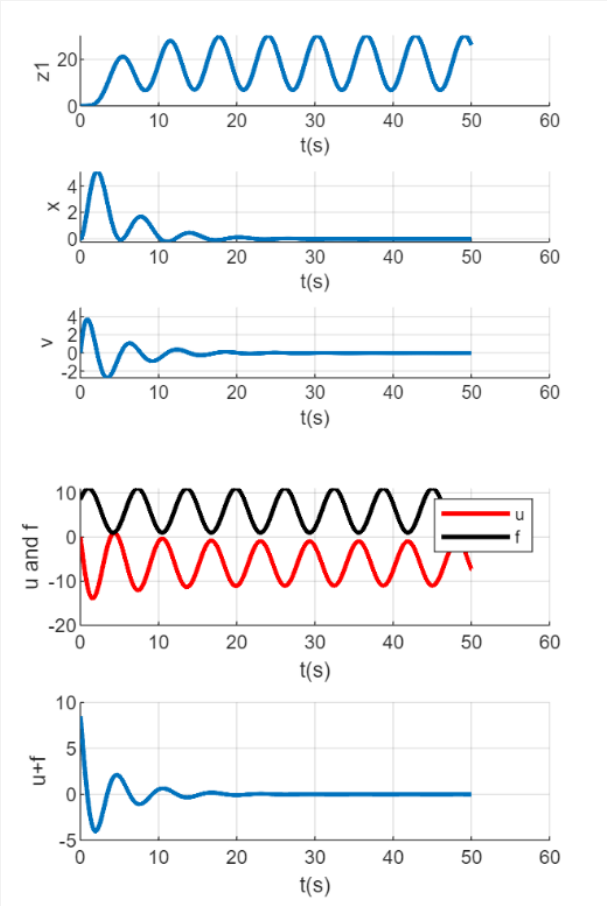
$\therefore \dot{z}=(A-Bk)z+Bf$

$$A=\begin{bmatrix}0&1&0&0\\ -\omega^2&0&1&0\\ 0&0&0&1\\ 0&0&0&0\end{bmatrix}\qquad B=\begin{bmatrix}0\\0\\0\\ \frac{1}{m}\end{bmatrix}$$

```
%% 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
```



4.2 火箭增程弹飞行仿真（while 循环+分段推力）

第 2 题（预先探究型题目）：

已知火箭增程弹的飞行力学方程如下

$$\begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \\ \dot{v} = -g\sin\theta - \dfrac{Q}{m} + P\cos\alpha \\ v\dot{\theta} = -g\cos\theta + \dfrac{Y}{m} + \dfrac{P\sin\alpha}{m} \end{cases} \quad (1)$$

其中 x, y 分别是火箭弹的飞行纵程和飞行高度，v, θ 分别是飞行速度大小和弹道倾角（弹道高低角），Q, Y 分别是气动阻力和气动升力，P 是火箭发动机推力，α 是火箭弹攻角（即弹体纵轴与飞行速度矢量间的夹角），g 是重力加速度，m 是火箭弹质量。

已知气动阻力和气动升力的计算公式为

$$\begin{cases} Q = \dfrac{1}{2}\rho v^2 S C_Q, \ Y = \dfrac{1}{2}\rho v^2 S C_Y \end{cases} \quad (2)$$

其中 ρ 是大气密度，S 是导弹气动特征面积（常参数），C_Q, C_Y 分别是气动阻力系数和气动升力系数。

请用 while 循环仿真如下飞行场景：

火箭弹以初速 v_0 和初始弹道倾角 θ_0 被推出发射筒后惯性爬升飞行 1 秒后，发动机点火，作动力飞行，发动机以恒定的推力 P 持续工作 10 秒后燃料耗尽，火箭弹继续惯性滑行，直至落地。

仿真入口参数如下：

$m = 100(\text{kg})$

$x(0) = 0,\ y(0) = 0,\ v_0 = 50(\text{m/s}),\ \theta_0 = 30^\circ$

$g = 9.8(\text{m/s}^2),\ \rho = 1.225(\text{kg/m}^3)$

$S = 0.01(\text{m}^2),\ C_Q = 1,\ C_Y = 0.2$

$\alpha = 3^\circ$（常值），$P = 500(\text{N})$

看看该弹射程多远？

并请绘制仿真结果曲线如下：

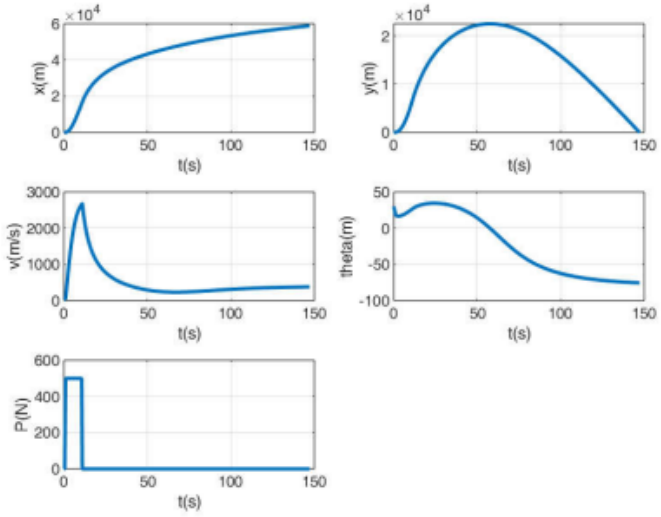


图 1 各状态变量和发动机推力曲线

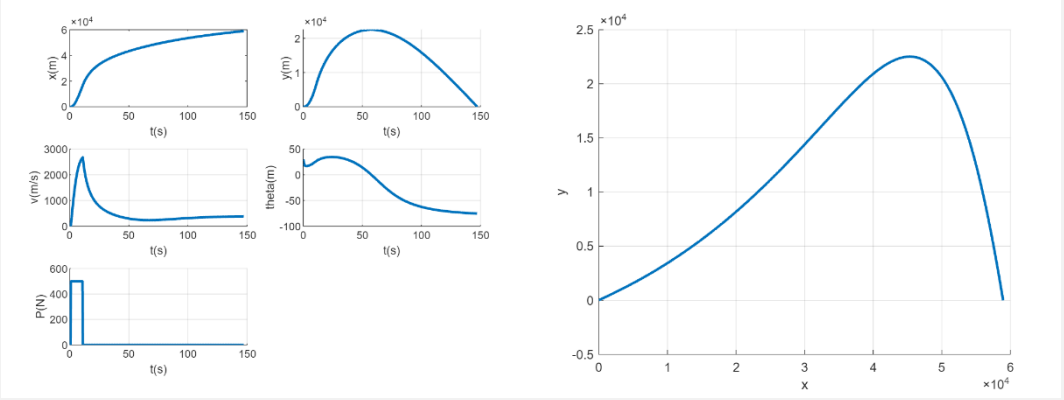
```
%% 参数设置
close all,clear,clc,

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
```

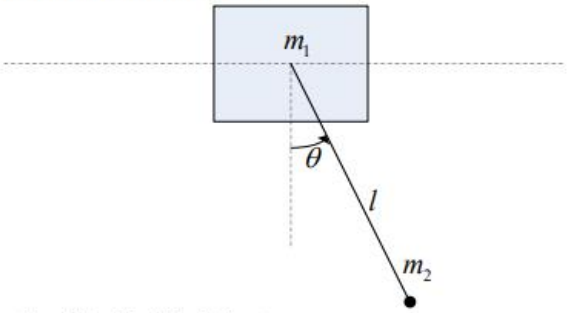


五、 耦合系统与制导率（滑块摆锤+导弹制导）

5.1 滑块——摆锤耦合系统非线性控制

第 1 题：

如下滑块和摆锤耦合系统



已知系统动力学方程如下

$$\begin{cases}(m_1+m_2)\ddot{x}+m_2 l\ddot{\theta}\cos\theta-m_2 l\dot{\theta}^2\sin\theta=F\\ m_2\ddot{x}\cos\theta+m_2 l\ddot{\theta}+m_2 g\sin\theta=0\end{cases}\quad(1)$$

其中：m_1, m_2 分别是滑块和摆锤的质量，l 是摆杆的长度，g 是重力加速度，x 是滑块的位置坐标，θ 是摆杆的摆角，F 是作用在滑块上的水平控制力。

现在设计了一个非线性控制律，以期实现滑块位置回零和摆杆摆角回零，控制律如下

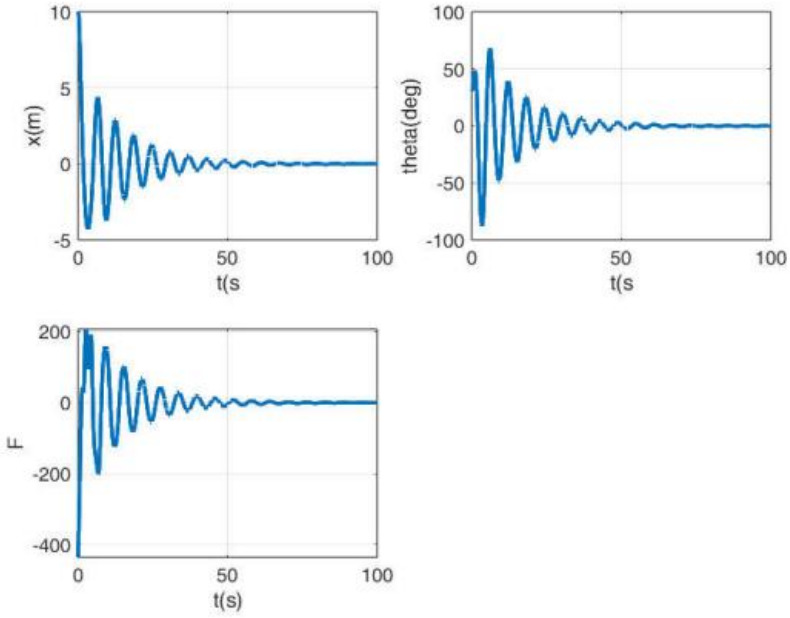
$$F=-m_2(g\cos\theta+l\dot{\theta}^2)\sin\theta-\\ k_p(m_1+m_2\sin^2\theta)(m_1 x-m_2 l\sin\theta)-k_d(m_1\dot{x}-m_2 l\dot{\theta}\cos\theta)\quad(2)$$

已知入口参数：

$$\begin{cases}m_1=20(\mathrm{kg}),\ m_2=10(\mathrm{kg}),\ l=5(\mathrm{m}),\ g=9.8(\mathrm{m/s^2})\\ x(0)=10(\mathrm{m}),\ \theta(0)=30^\circ,\ \dot{x}(0)=0,\ \dot{\theta}(0)=0\end{cases}$$

试自编龙格库塔程序对系统的受控运动过程进行仿真。

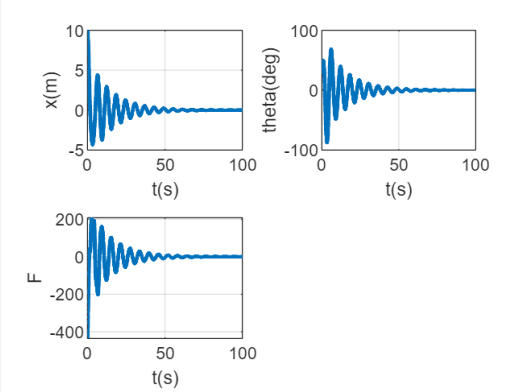
仿真效果如下图：




```
close all,clear,clc,
tic,
%% 参数设置
```

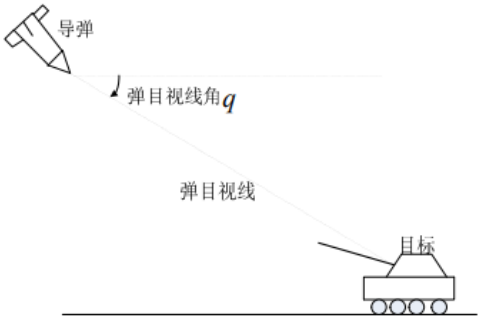
```
global m1 m2 l g
m1 = 20;
m2 = 10;
l = 5;
g = 9.8;

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
```



5.2 导弹制导率仿真（比例导引+升力限幅）

第 2 题：



已知导弹运动简化方程如下

$$\begin{cases}\dot{x}_m = v_m\cos\theta_m \\ \dot{y}_m = v_m\sin\theta_m \\ \dot{v}_m = -g\sin\theta_m - \dfrac{Q}{m} \\ v_m\dot{\theta}_m = -g\cos\theta_m + \dfrac{Y}{m}\end{cases}$$

其中 x_m 表示导弹纵程，y_m 表示其飞行高度，v_m 表示导弹速度，θ_m 表示弹道倾角，m 表示导弹质量，g 表示当地重力加速，$Q=\dfrac{1}{2}\rho v_m^2 SC_Q$ 表示导弹受到的气动阻力，ρ 表示当地大气密度，S 表示导弹气动特征面积，C_Q 表示导弹气动阻力系数（由空气动力学计算或风洞试验测定），Y 表示导弹受到的气动升力。

目标因为是地面的，且为简化计，只考虑其纵向运动，故其运动学方程为

$$\dot{x}_t = v_t$$

现设导弹制导律的形式已经设计好：

$Y = kmv_m\dot{q} + mg\cos\theta_m$（带重力补偿的变系数自适应比例导引律）

并且考虑到实际情况，应对 Y 进行限幅，即 Y 的绝对值不能超过 $20mg$。

其中 q 表示弹目视线角，$\dot{q}$ 表示弹目视线角速率，其表达式分别如下

$$q = -\arctan\frac{y_m}{x_t - x_m},\quad \dot{q} = \frac{v_m}{r}\sin(q-\theta_m) - \frac{\dot{x}_t}{r}\sin q$$

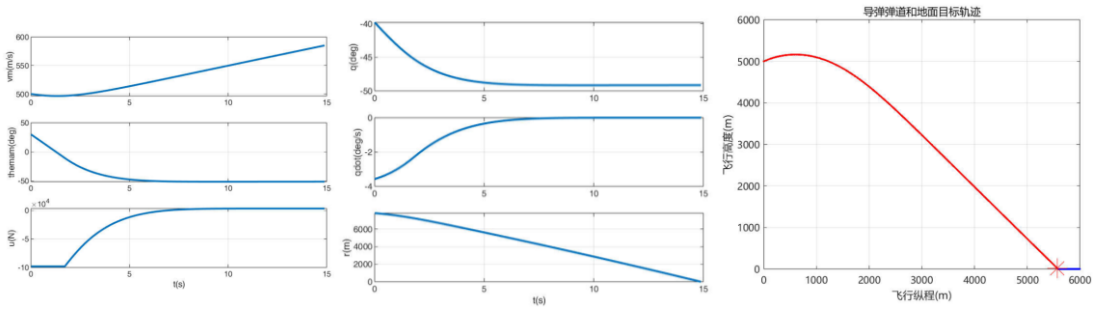
其中 r 表示弹目距离，可通过导引头在线测量。

仿真参数：

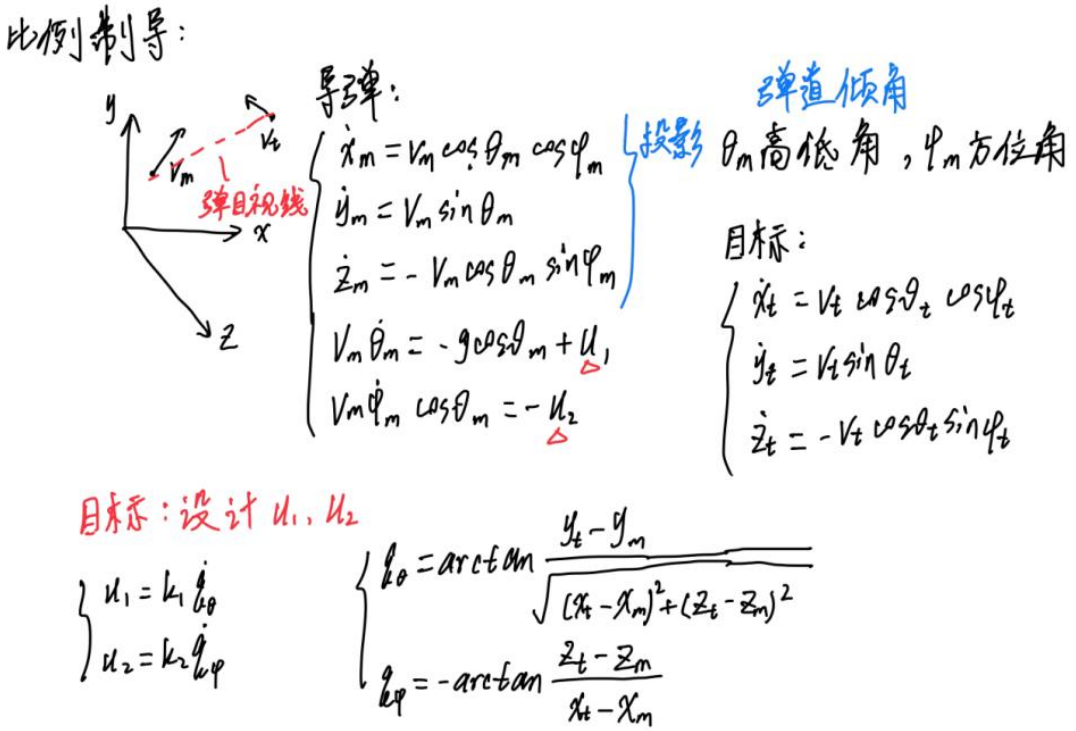
$m=500(\text{kg})$，$x_m(0)=0(\text{m})$，$y_m(0)=5000(\text{m})$，$v_m(0)=500(\text{m/s})$，$\theta_m(0)=0^o$（即水平发射），$x_t(0)=6000(\text{m})$，$v_t\equiv -72(\text{km/h})$（目标匀速直线运动），$\rho=1.225(\text{kg/m}^3)$，$g=9.8(\text{m/s}^2)$，$S=0.01(m^2)$，$C_Q=0.1$.

请自编龙格库塔仿真程序，验证制导律的有效性。

仿真效果图如下：



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];
```

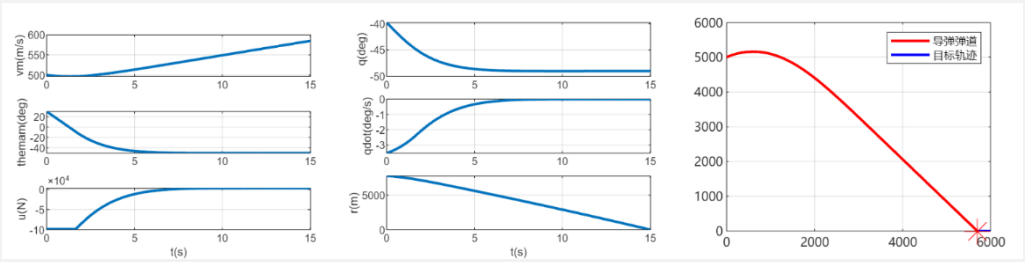
```
stateout = state;

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
```



六、 模板

6.1 非线性系统与数值仿真基础

6.1.1 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
```

6.1.2 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
```

6.2 PID/PD 控制设计模板

6.2.1 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
```

6.2.2 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）
```

6.3 倒立摆系统控制（双通道 PD + 状态反馈）

6.3.1 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
```

6.3.2 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
```

6.4 复杂干扰与运动轨迹（复合干扰控制 + 火箭飞行）

6.4.1 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
```

6.4.2 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
```

6.5 耦合系统与制导律（滑块摆锤 + 导弹制导）

6.5.1 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
```

6.5.2 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
```

6.6 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');
```