南京航空航天大学

2025－2026 学年 第1学期大作业

《飞行器建模与分析》

大作业设计报告

2025年10月

目录

[一、 基础题 3](#_Toc211442530)

[1.1 基础题一 3](#_Toc211442531)

[1.2 基础题二 4](#_Toc211442532)

[1.3 基础题三 6](#_Toc211442533)

[1.4 基础题四 7](#_Toc211442534)

[二、 应用题 10](#_Toc211442535)

[2.1 应用题五 10](#_Toc211442536)

[2.2 应用题六 11](#_Toc211442537)

[2.3 应用题七 13](#_Toc211442538)

[三、 综合题 15](#_Toc211442539)

[3.1 综合题八 15](#_Toc211442540)

[3.1.1 位置方程组建模 16](#_Toc211442541)

[3.1.2 速度方程组建模 17](#_Toc211442542)

[3.1.3 气流姿态角方程组 17](#_Toc211442543)

[3.1.4 角速率方程组 18](#_Toc211442544)

[3.1.5 模型封装 19](#_Toc211442545)

[3.2 真模型 20](#_Toc211442546)

[3.2.1 代码补全 20](#_Toc211442547)

[3.2.2 绘图程序 22](#_Toc211442548)

[3.2.3 结果 24](#_Toc211442549)

[3.3 综合题九 24](#_Toc211442550)

[3.3.1 情形一 25](#_Toc211442551)

[3.3.2 情形二 25](#_Toc211442552)

[3.3.3 情形一和二最小速度求解及结果对比 26](#_Toc211442553)

[3.4 综合题十 29](#_Toc211442554)

[3.4.1 精确模型建模 29](#_Toc211442555)

[3.4.2 简易模型建模 30](#_Toc211442556)

[3.4.3 复杂模型与简易模型对比 32](#_Toc211442557)

[附录 33](#_Toc211442558)

# 基础题

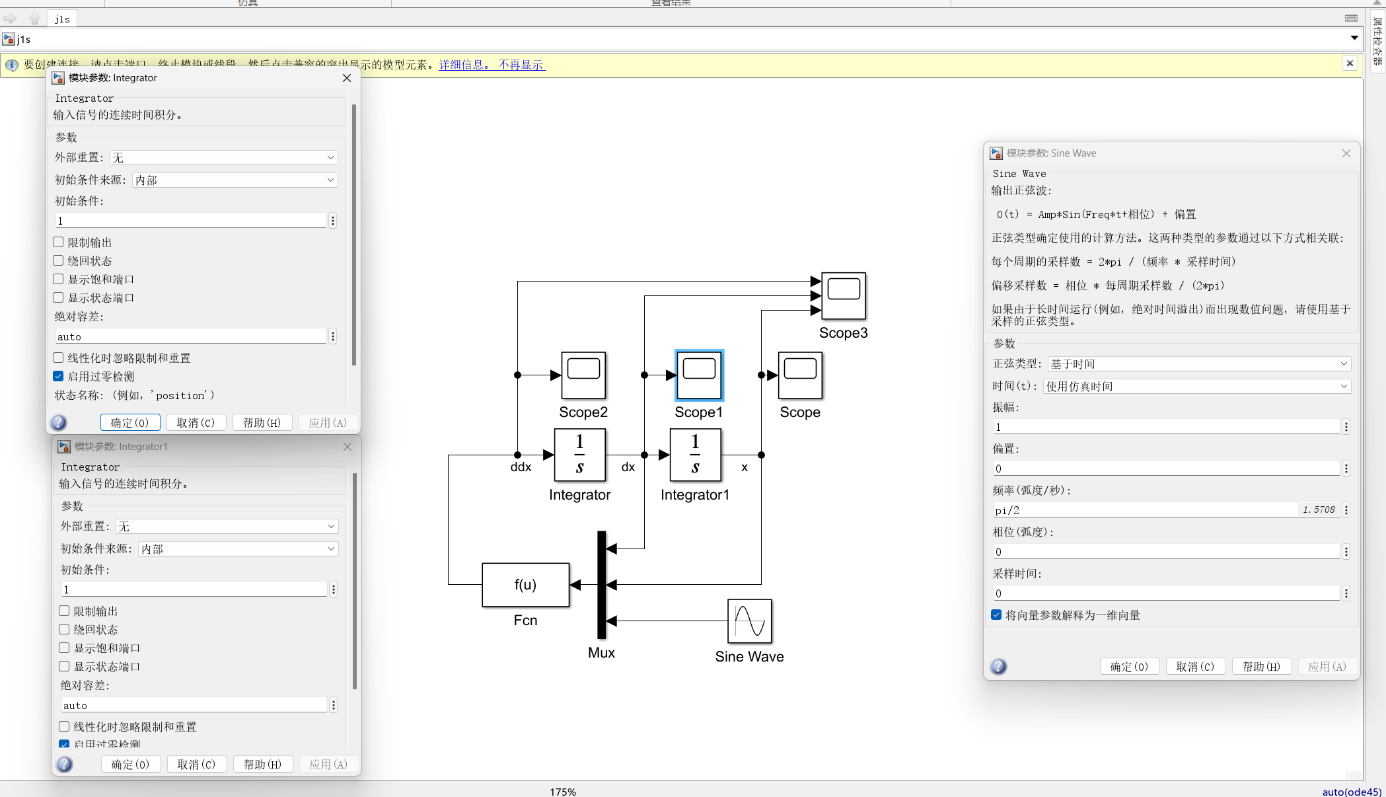
## 基础题一

分析下列方程式所示的系统行为



初始条件为，用MATLAB/SIMULINK给出仿真结果。

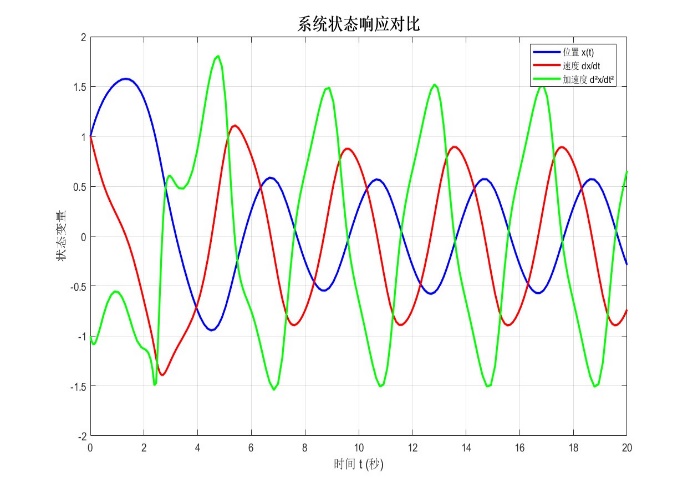
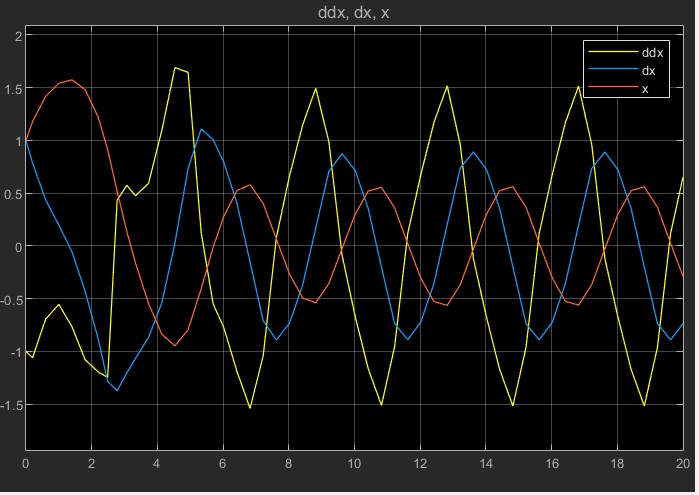
根据题目构建如下系统：



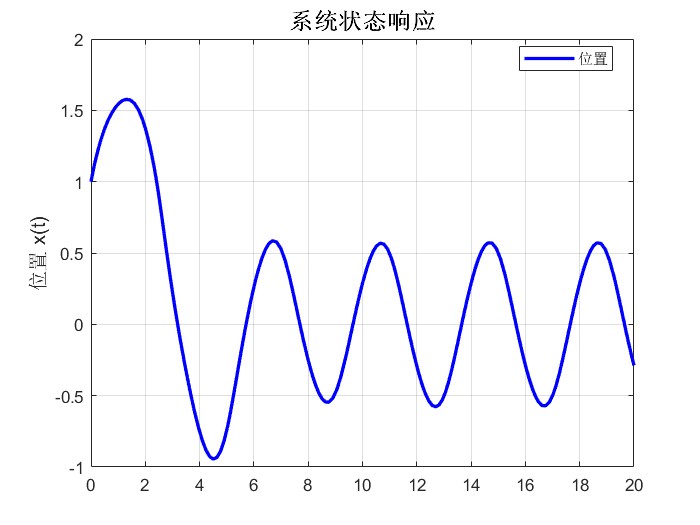
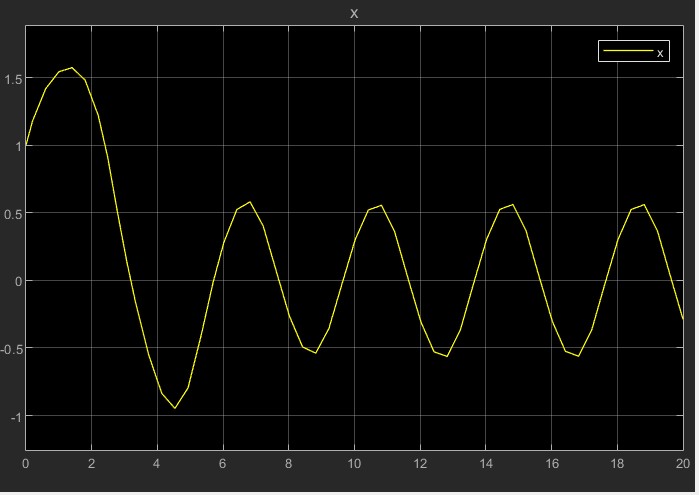
问题1SIMULINK建模

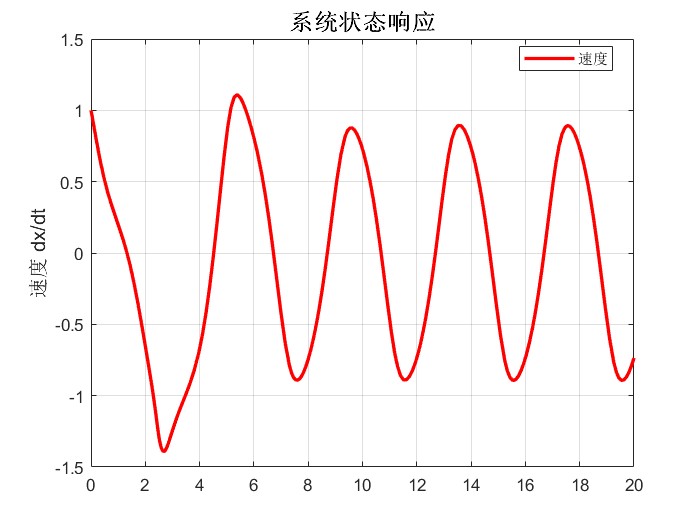
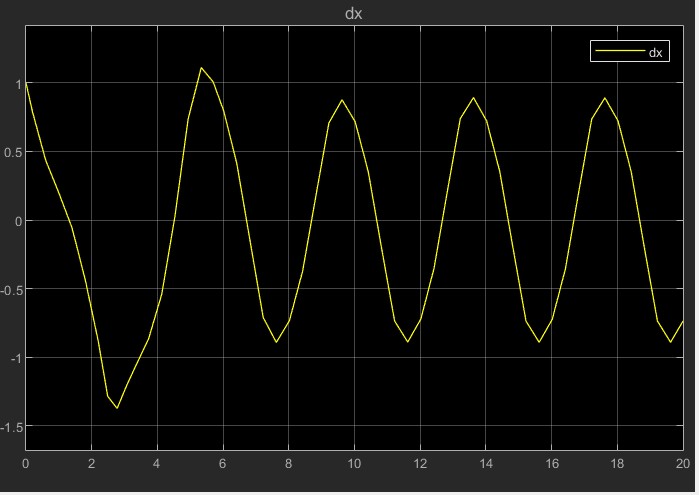
同时运用matlab程序进行求解结果对比，程序源码详见附录。

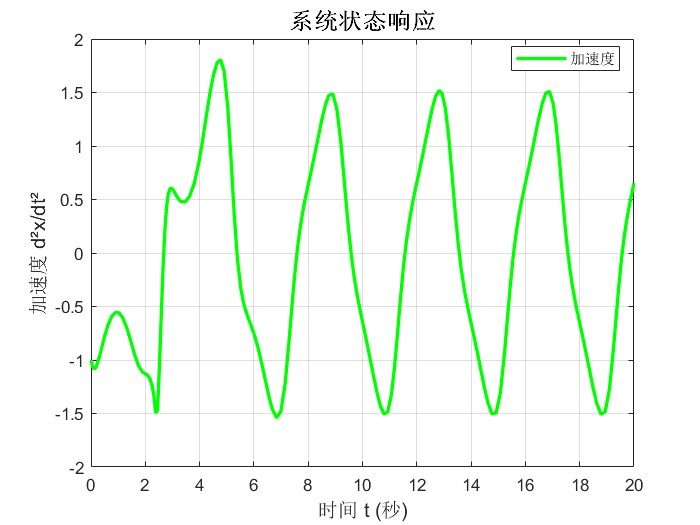
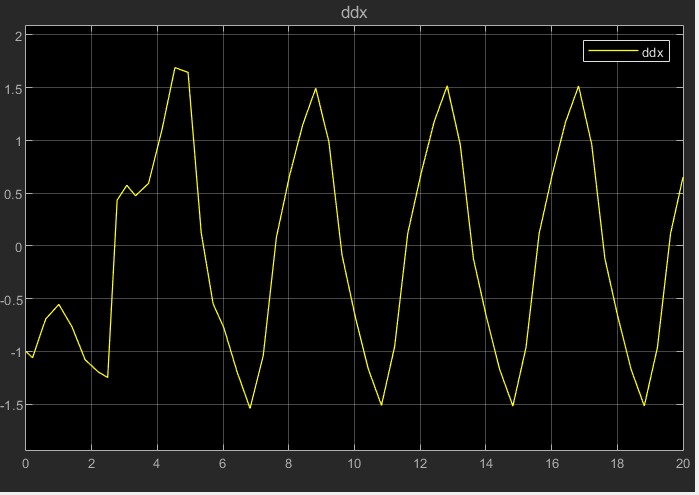
结果如下图所示（左侧图为SIMULINK仿真结果，右侧图为MATLAB程序运行结果）：



此图为对比图。







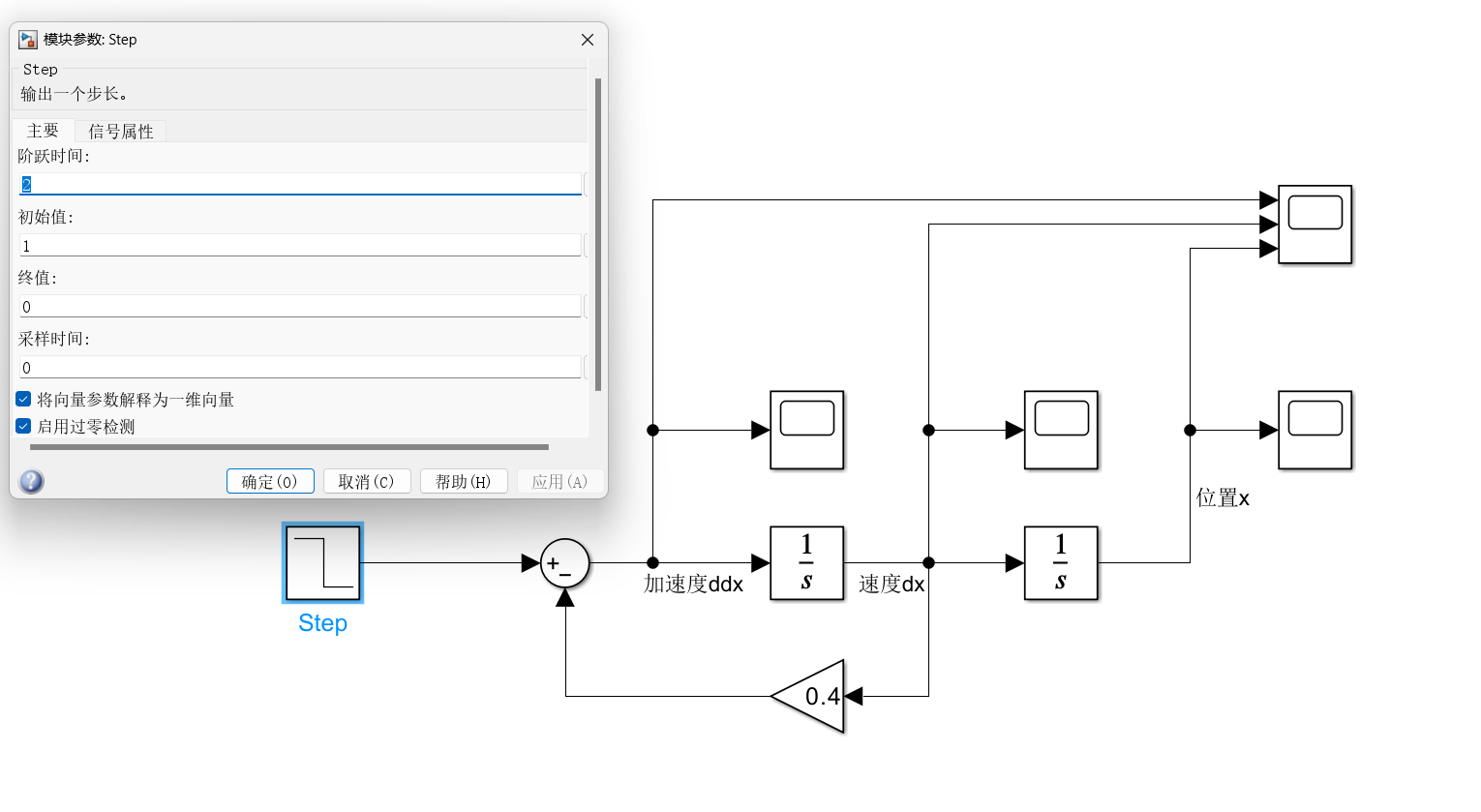
## 基础题二

力-质量系统，要拉动一个箱子，箱子质量为 ，箱子与地面存在摩擦力，其大小与车子的速度成正比。其运动方程式为



其中，拉力作用时间为，用MATLAB/SIMULINK给出仿真结果。

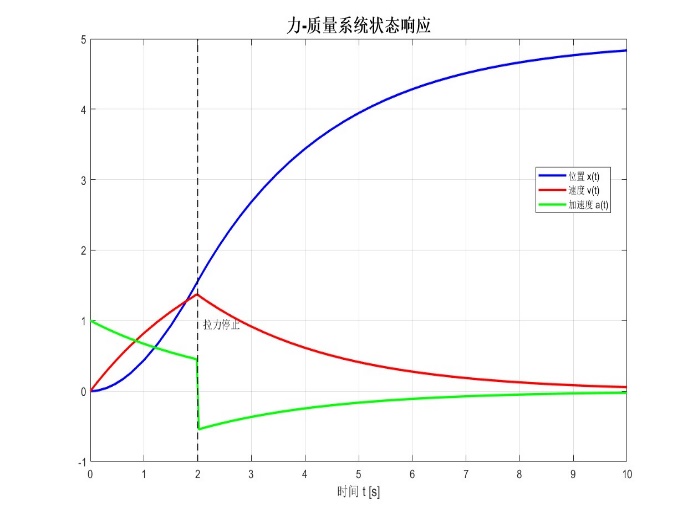
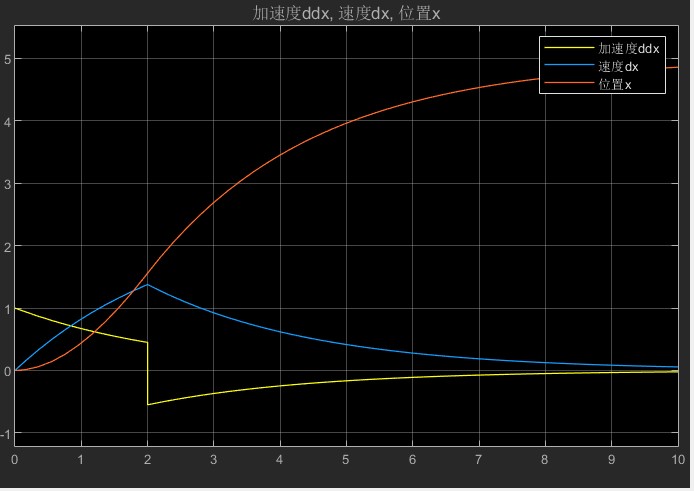
根据题目构建如下系统：

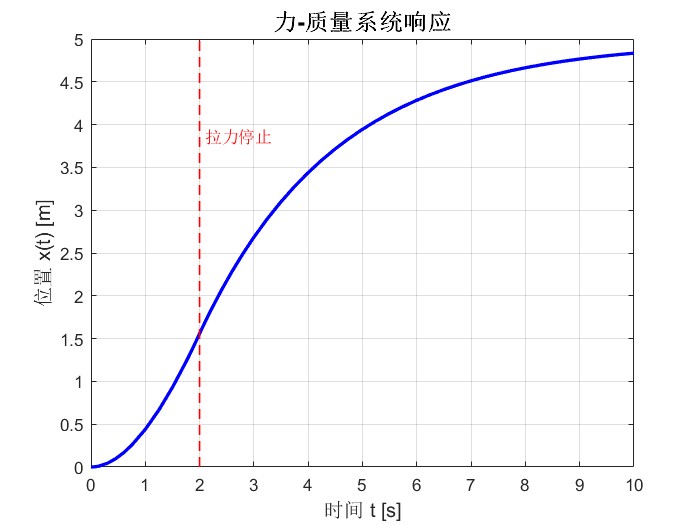
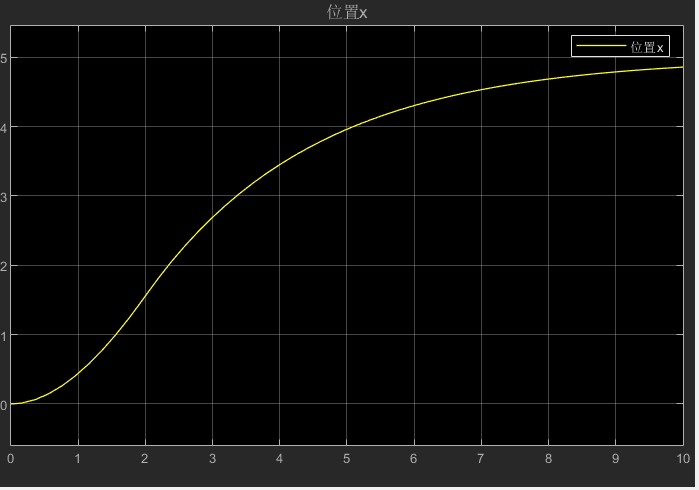


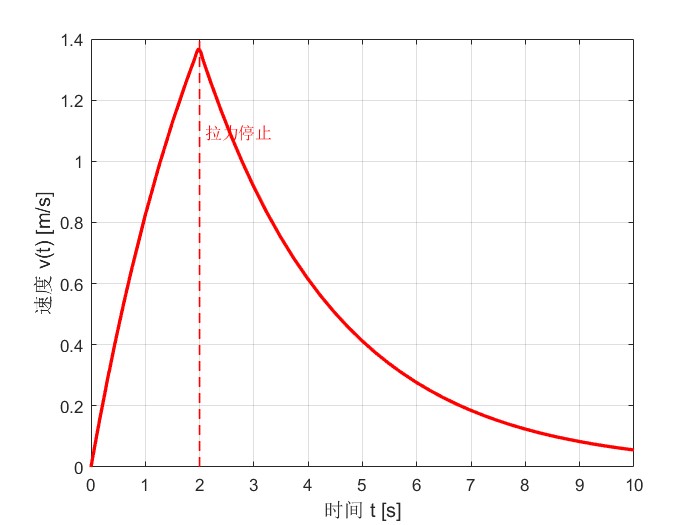
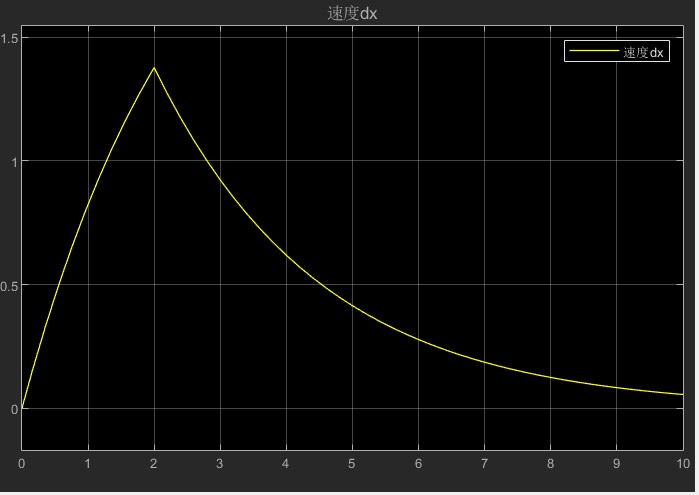
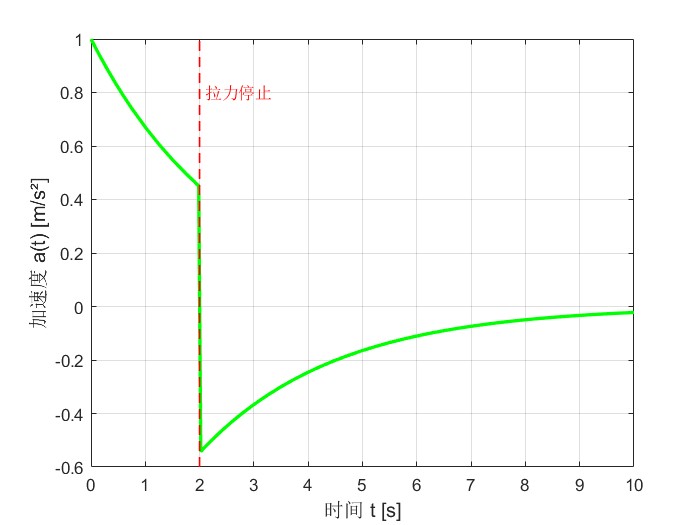
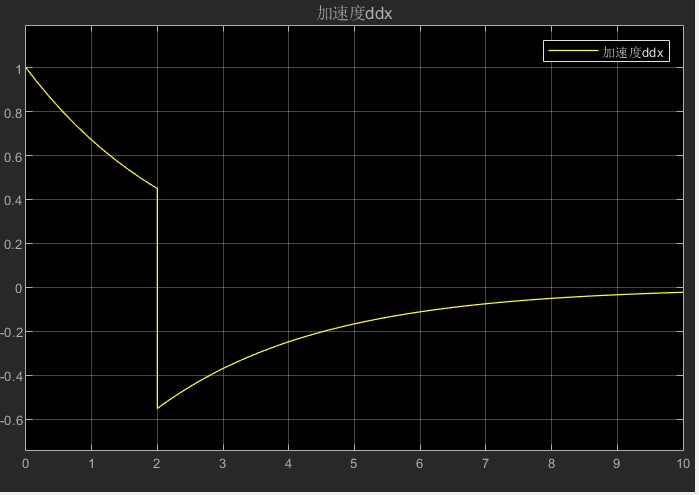
问题2SIMULINK建模

同时运用matlab程序进行求解结果对比，程序源码详见附录。

结果如下图所示（左侧图为SIMULINK仿真结果，右侧图为MATLAB程序运行结果）：





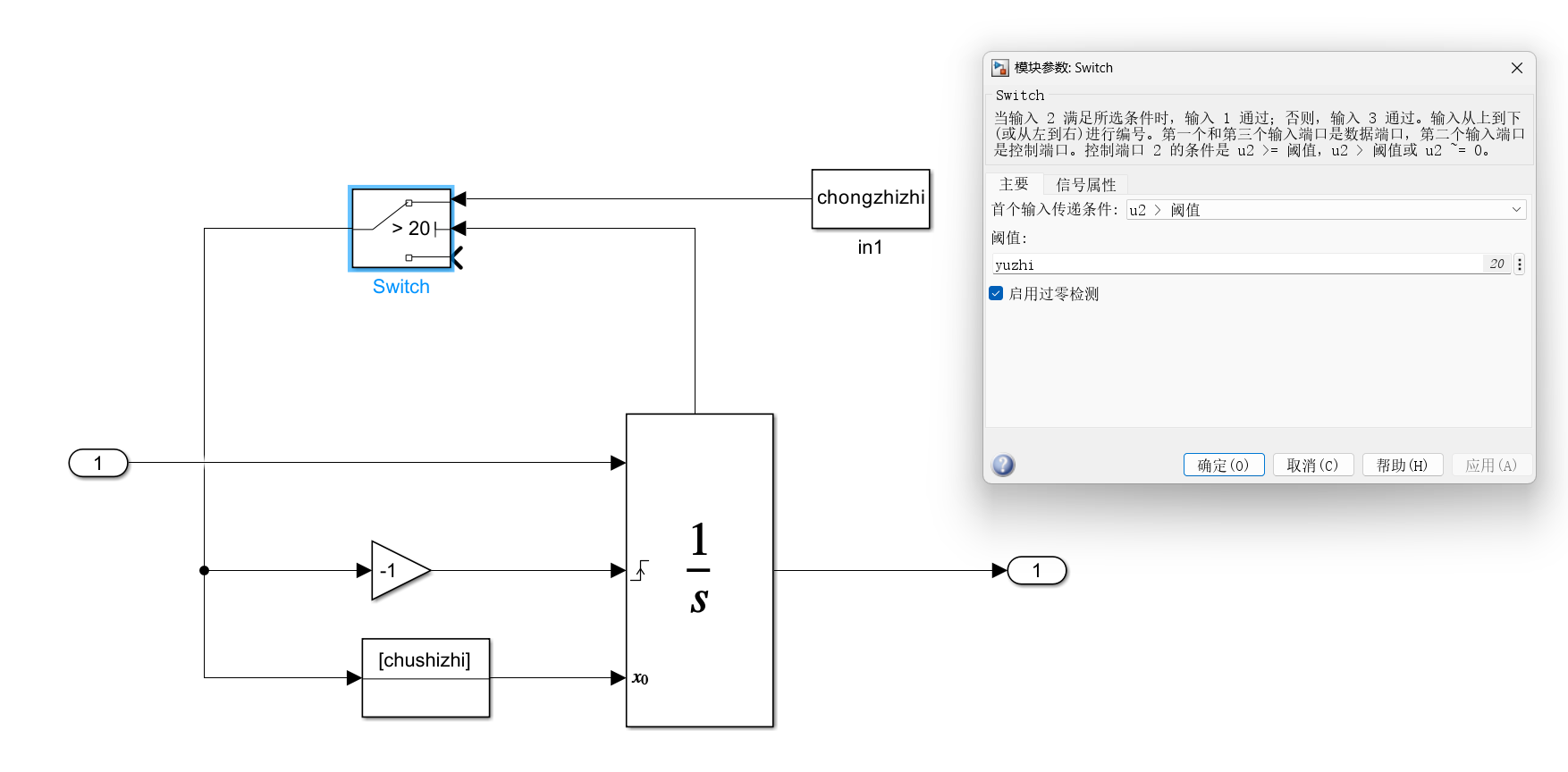
## 基础题三

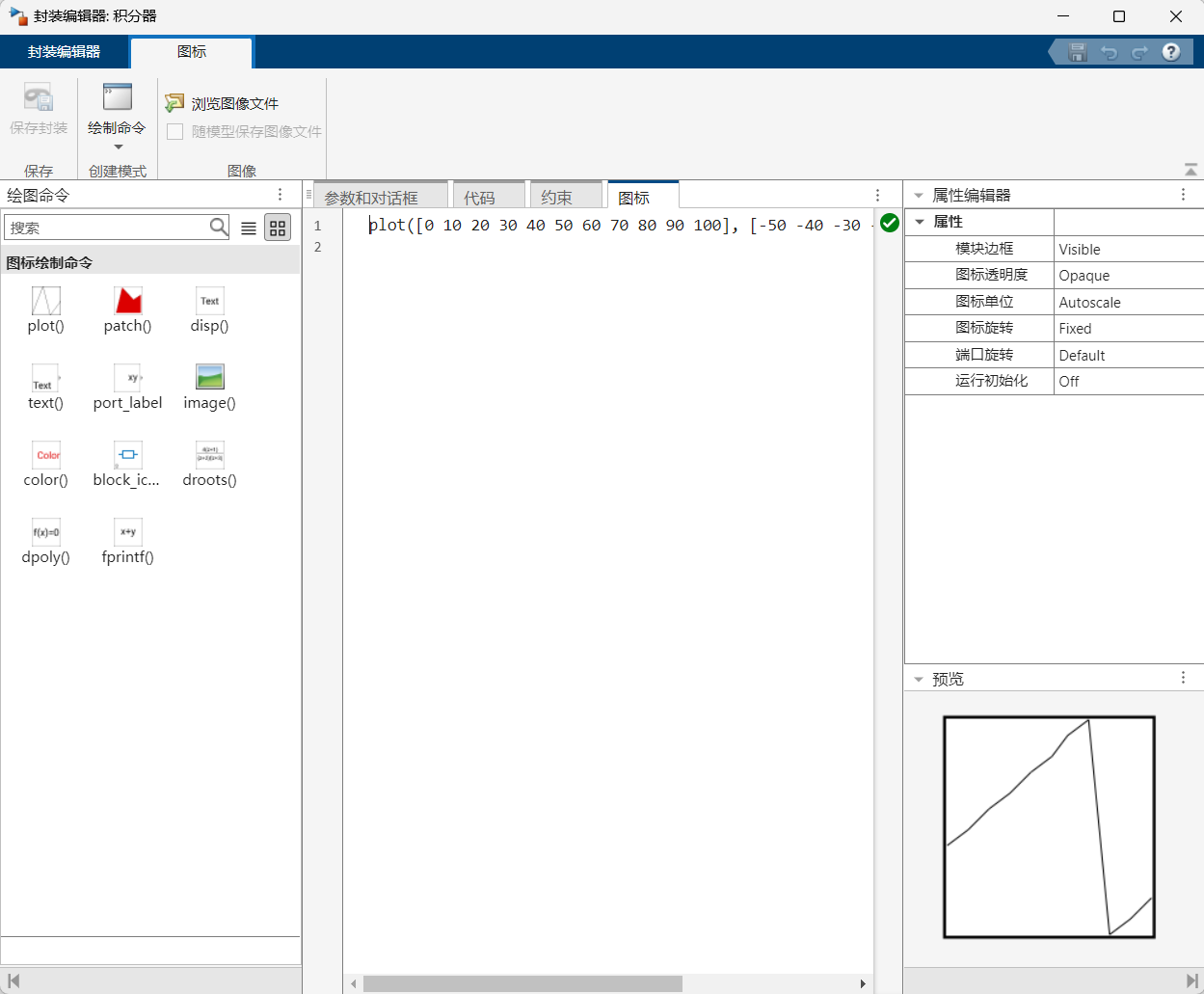
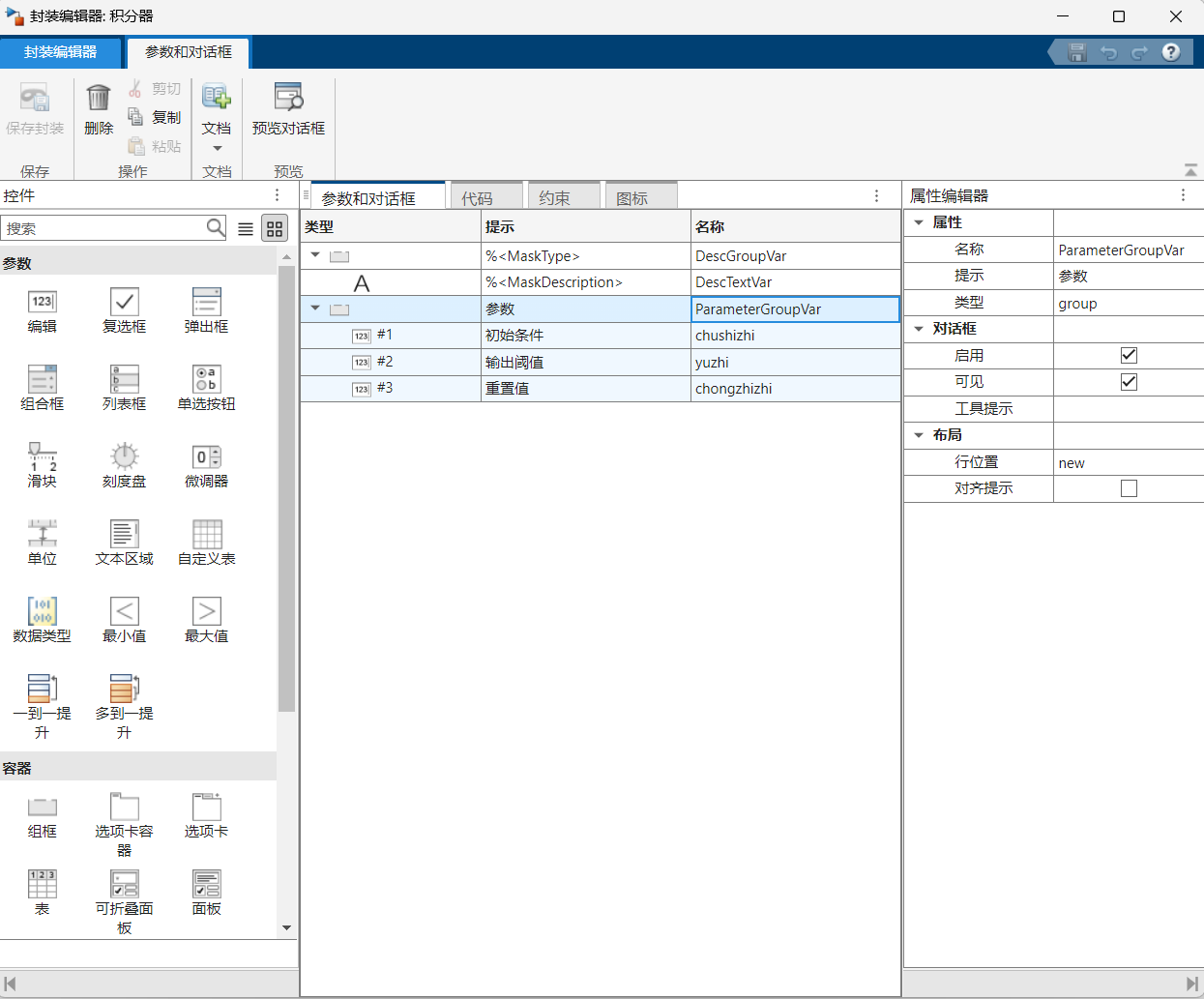
建立一个积分器，输入为1，初始条件为-50，如果输出超过20，则重置为-100，用MATLAB/SIMULINK给出仿真结果。

根据题目构建如下系统：

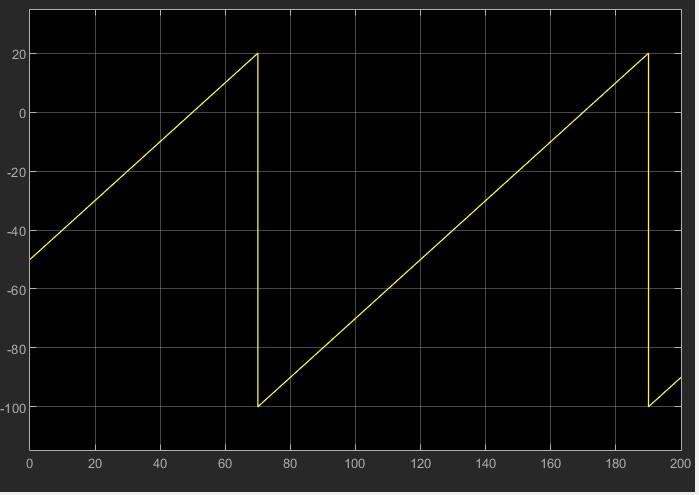
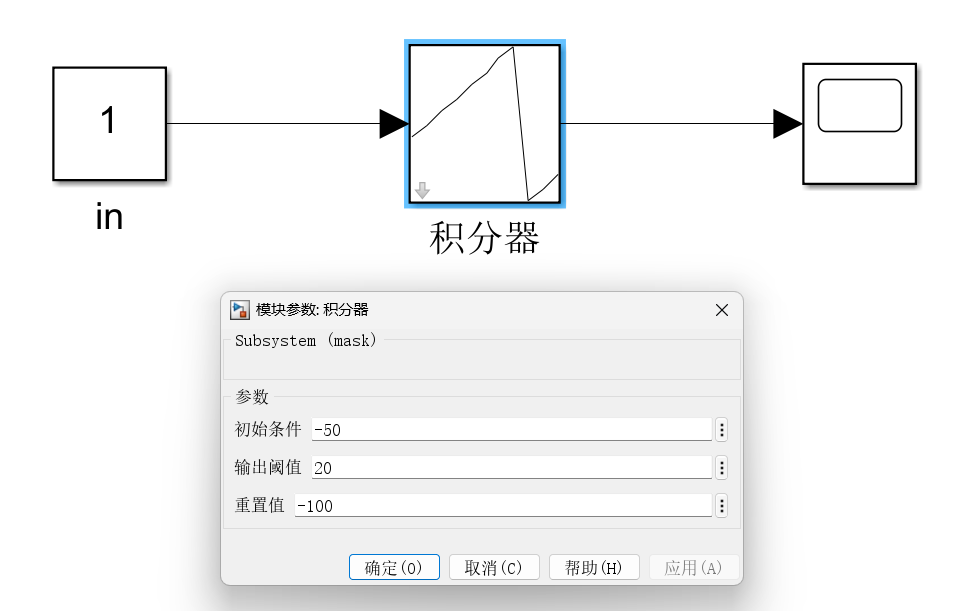


将题目要求中的初始条件、输出阈值以及重置值作为未知量对子系统进行封装，如下所示：





最终得到如下所示的SIMULINK模型以及结果如图所示：



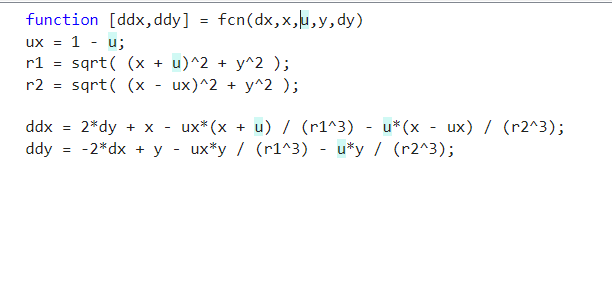
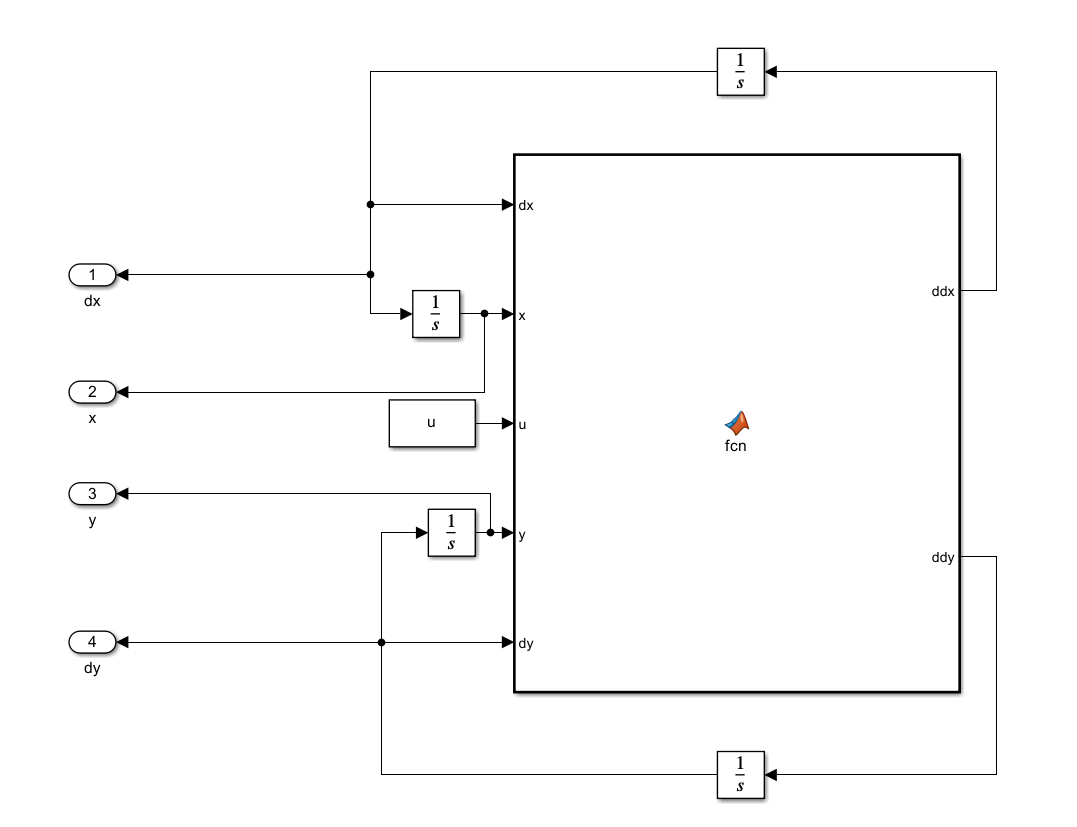
## 基础题四

已知某空间飞行器的运动轨迹满足下面的方程

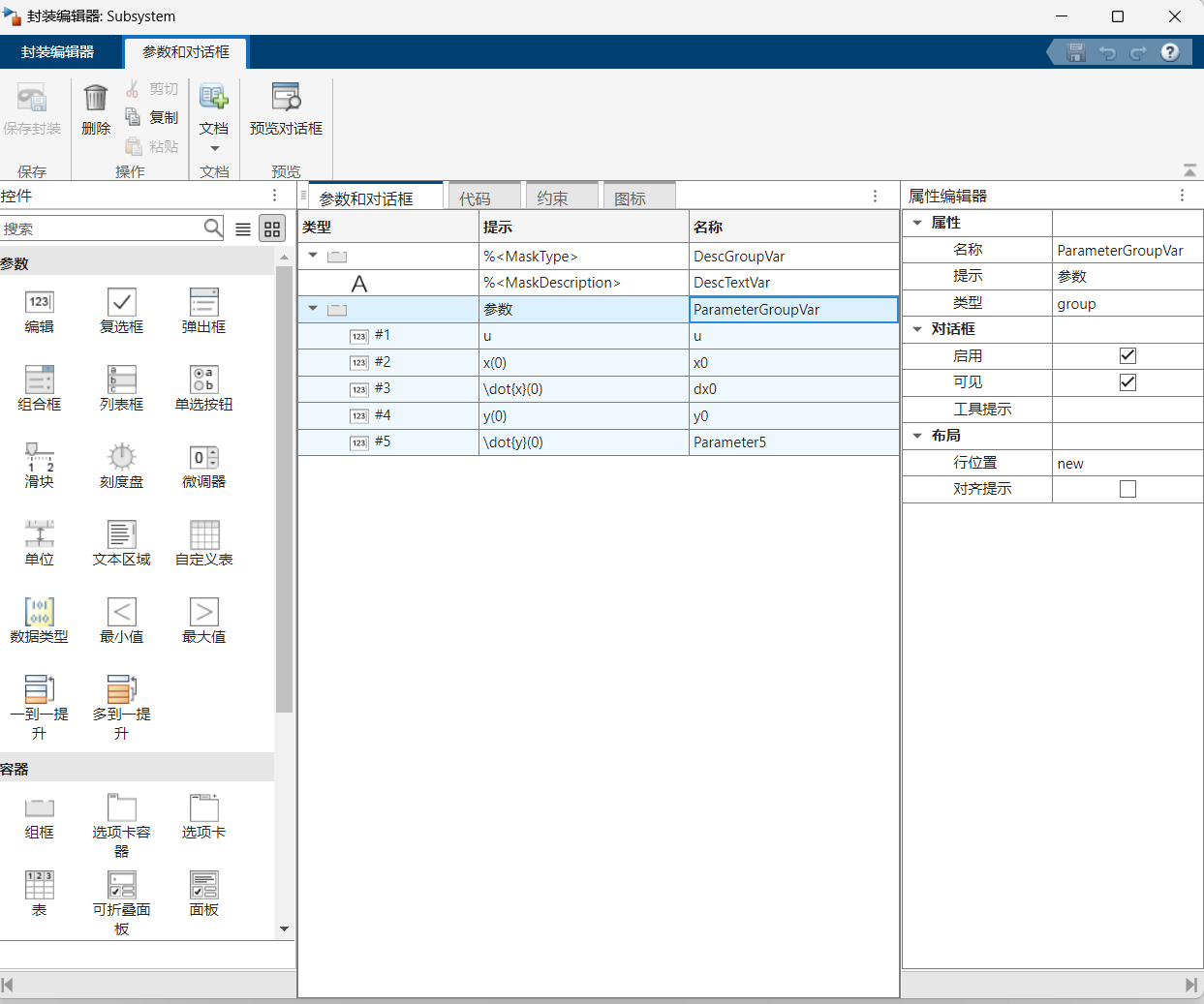
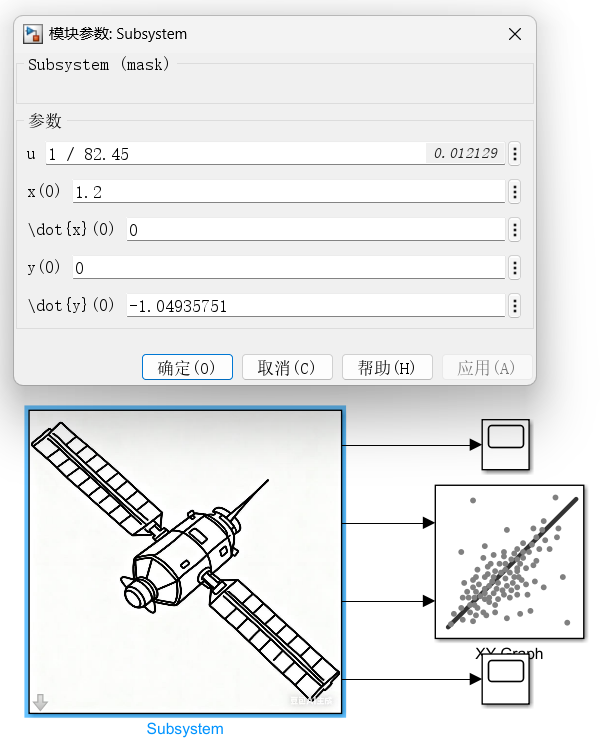


其中，，试在初值下用MATLAB/SIMULINK进行求解，并绘制出某空间飞行器位置的轨迹。

根据题目构建如下系统：

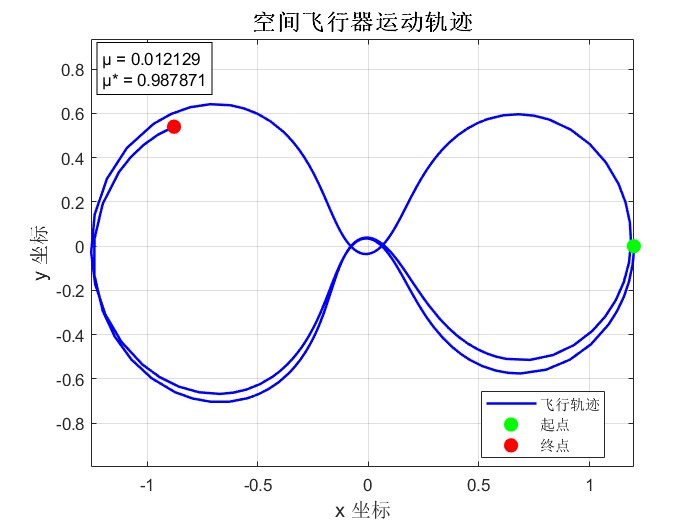
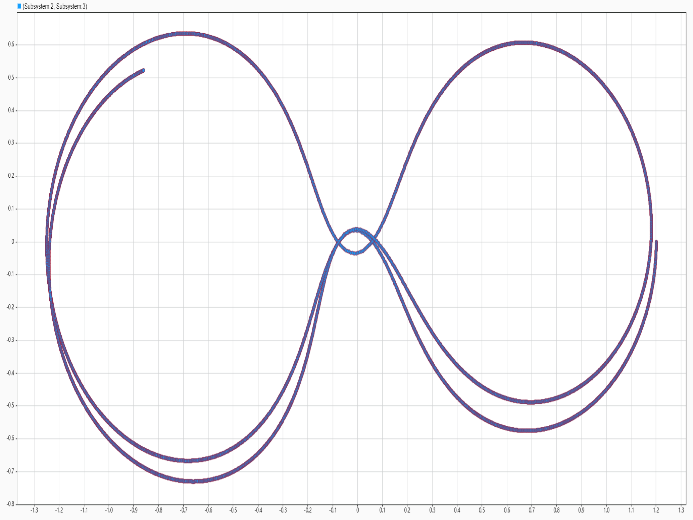


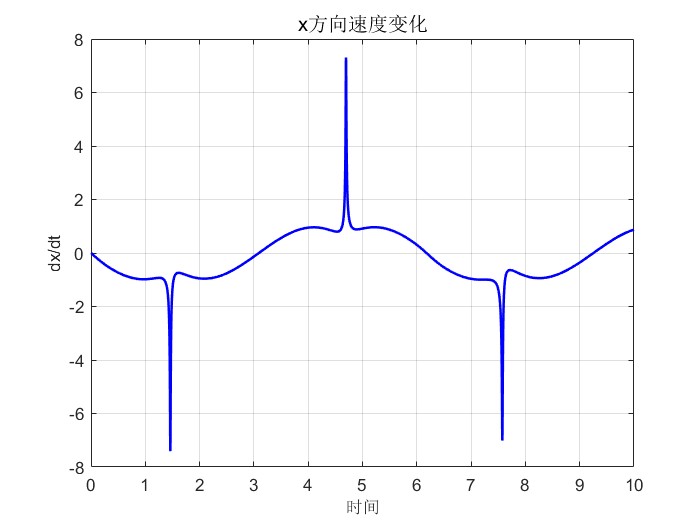
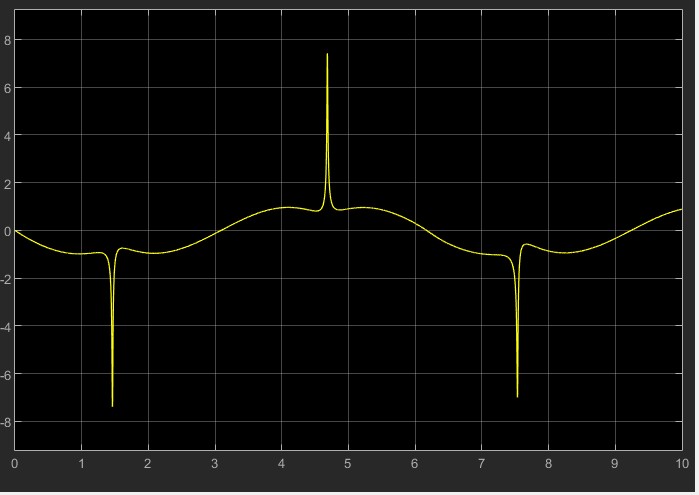
将题目要求中作为未知量对子系统进行封装，如下所示：

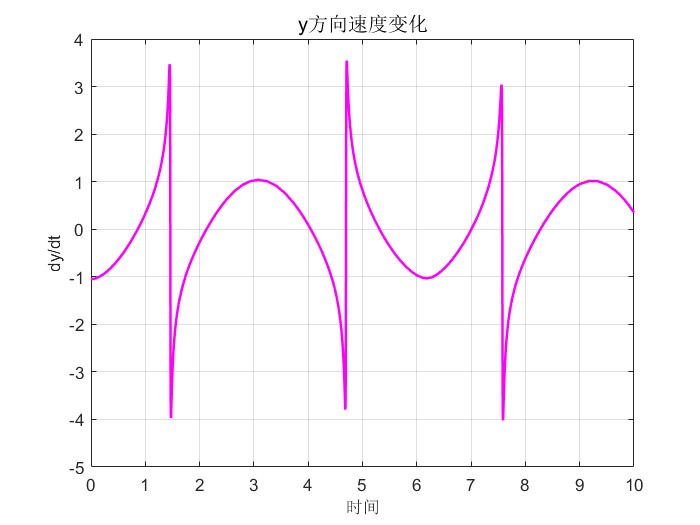
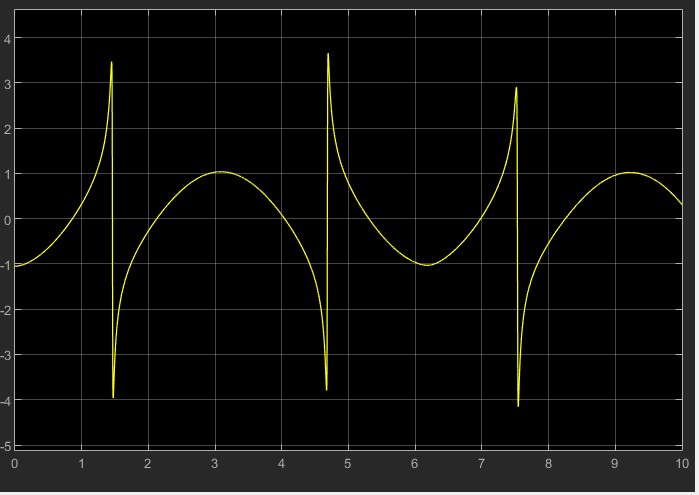


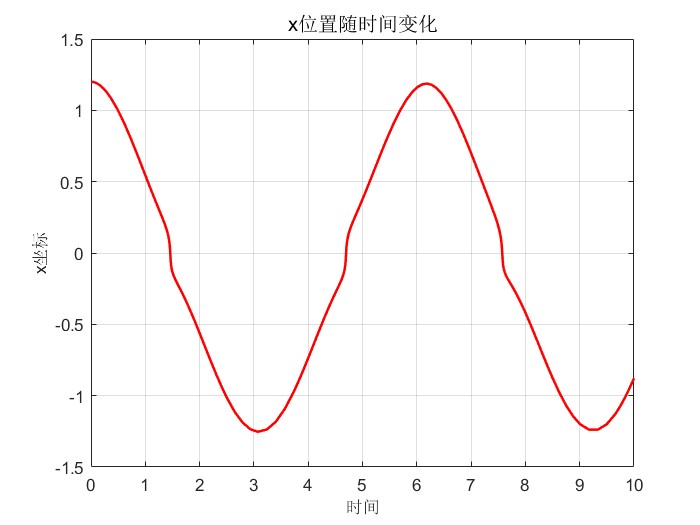
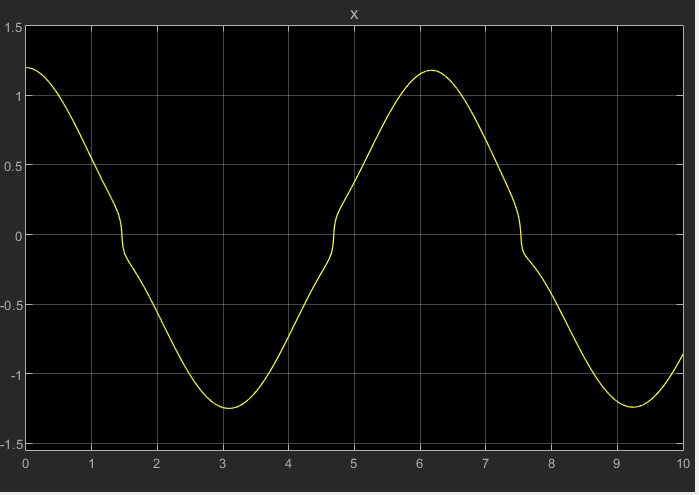
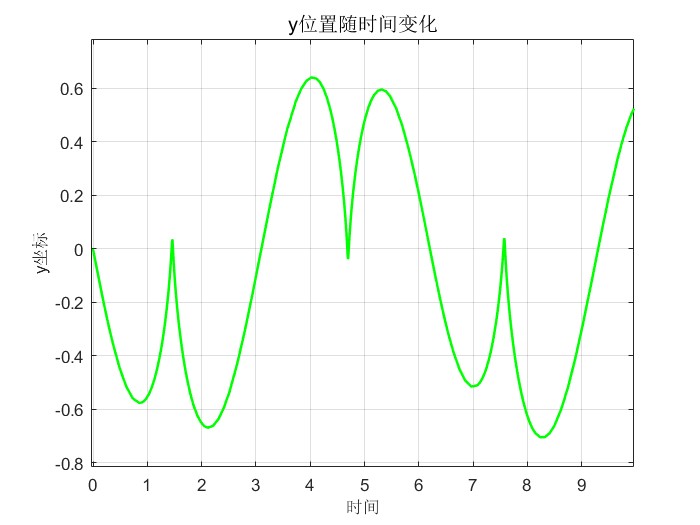
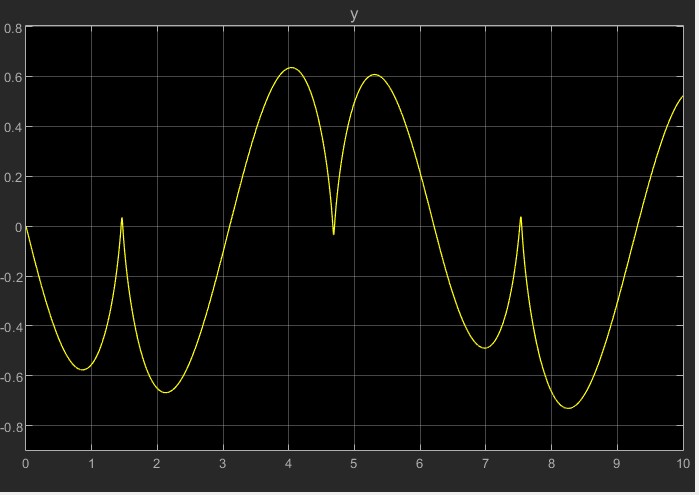
同时运用matlab程序进行求解结果对比，程序源码详见附录。

结果如下图所示（左侧图为SIMULINK仿真结果，右侧图为MATLAB程序运行结果）：







# 应用题

## 应用题五

蹦极者质量为 m，设桥梁与水面之间的距离为80m，弹性绳长为30m，刚度系数为k，一端系在桥梁上，另一端系于蹦极者。从桥梁位置开始无初速度落下，坐标原点O取在绳长位置。

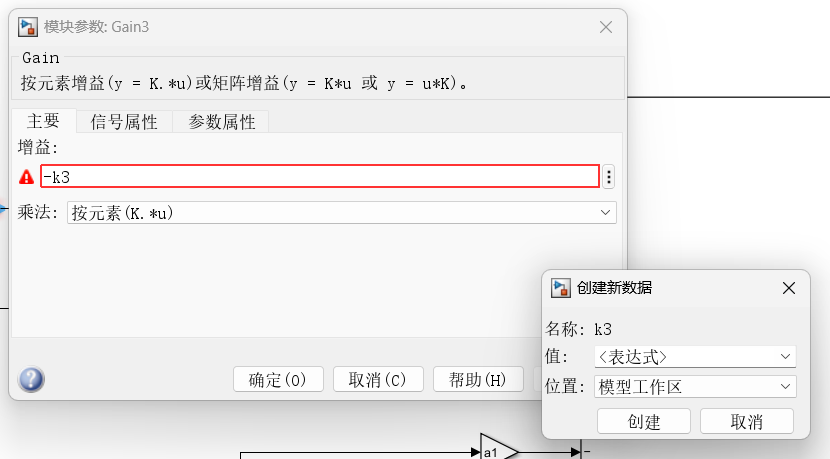


其中，

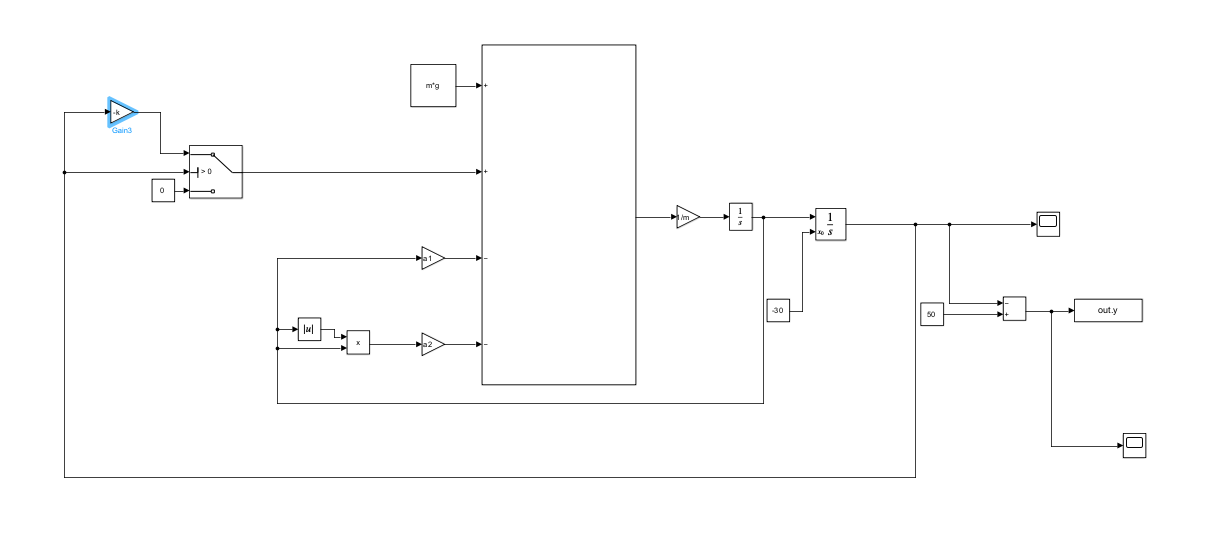


其中。若蹦极者最大质量为100 kg，用MATLAB/SIMULINK给出蹦极安全的弹性绳最小刚度系数k。

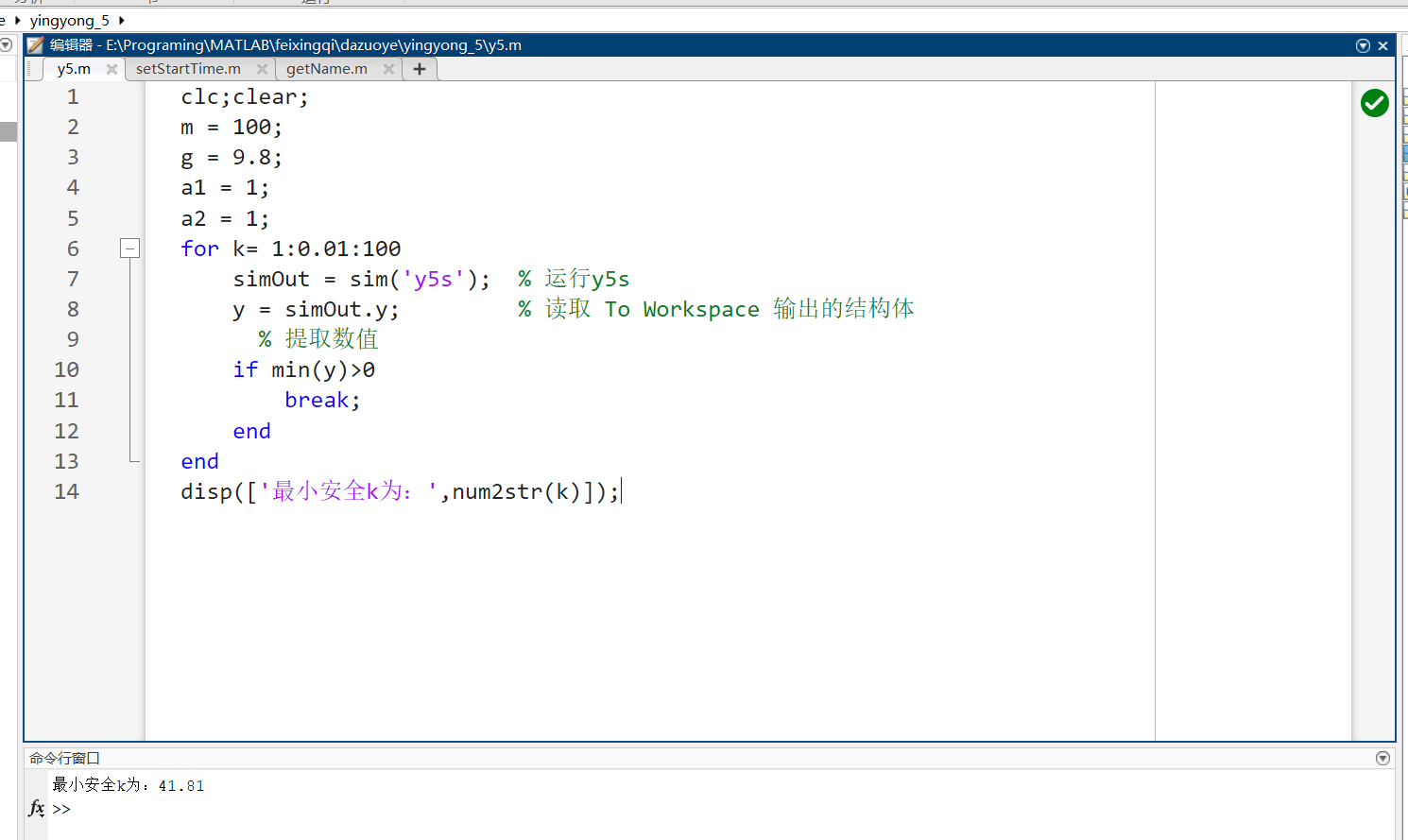
此处首先构建蹦极模型，接着将刚度系数k设置为模型工作区变量，如下图k3所示，接着便可在MATLAB工作区对k进行修改便可通过简单遍历来确定最小刚度系数k。



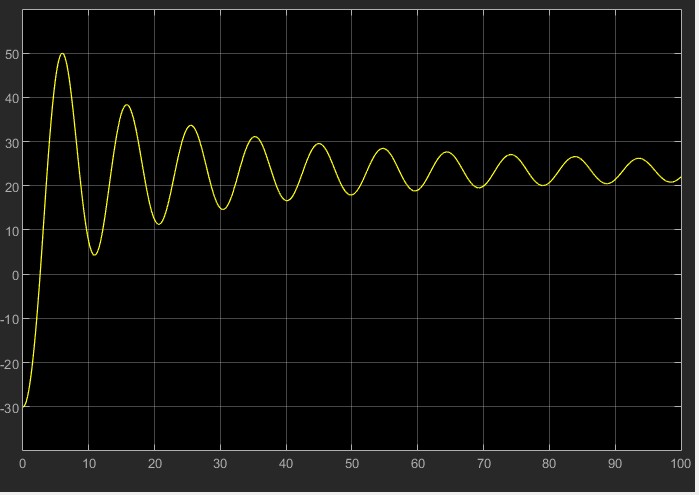
接着需要在构建的模型中将人的实际下落高度通过To workspace模块传输到工作区，从而通过判断位置的最小值是否大于0来判断是否是最小刚度系数k。



由此，k的遍历便可以由以下工作区脚本代码实现。



最终运行结果如下图所示，最小安全弹簧刚度k为41.81；



## 应用题六

假设著名的Lorenz模型的状态方程表示为

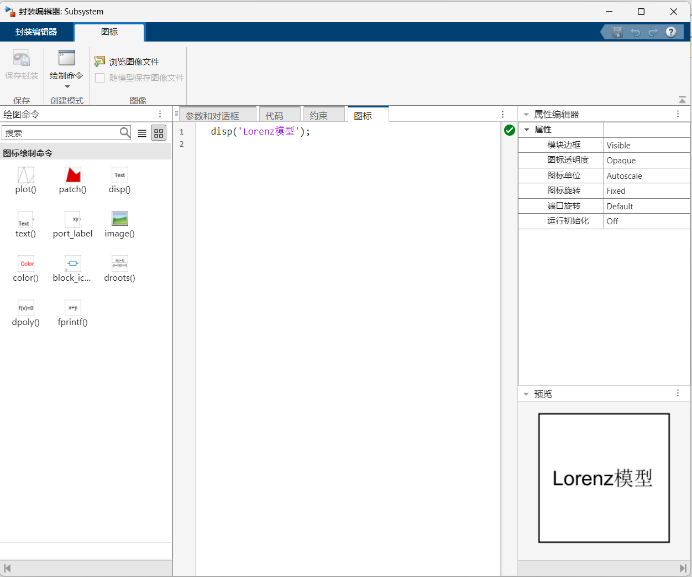
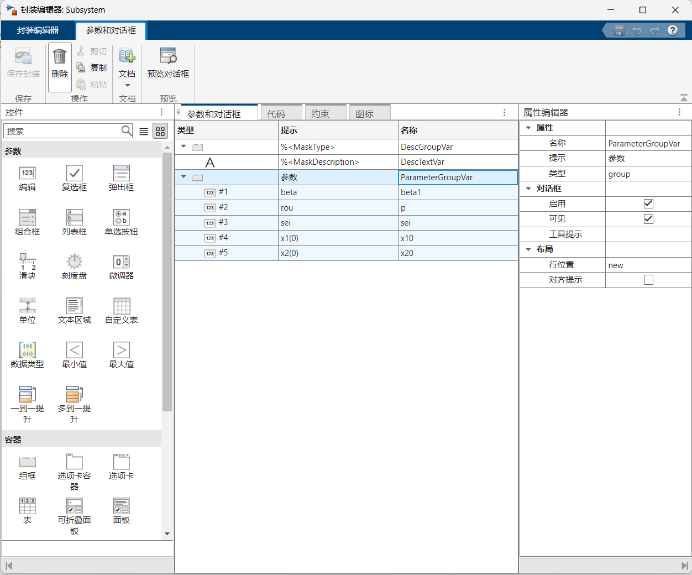


其中。若另其初值为，而为机器上可以识别的小常数，如取一个很小的正数，用MATLAB/SIMULINK求解该微分方程。

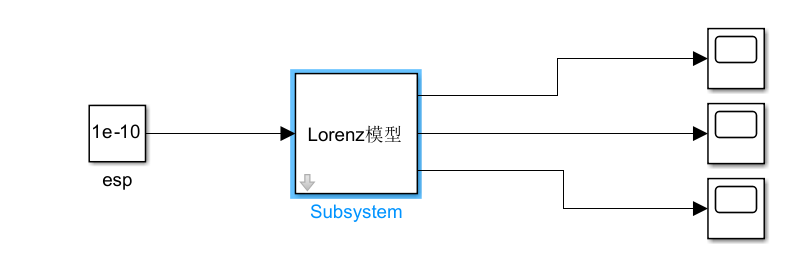
根据题目构建如下系统：



接着以为未知量对系统进行封装，同时作为子系统输入量，作为子系统输出量：

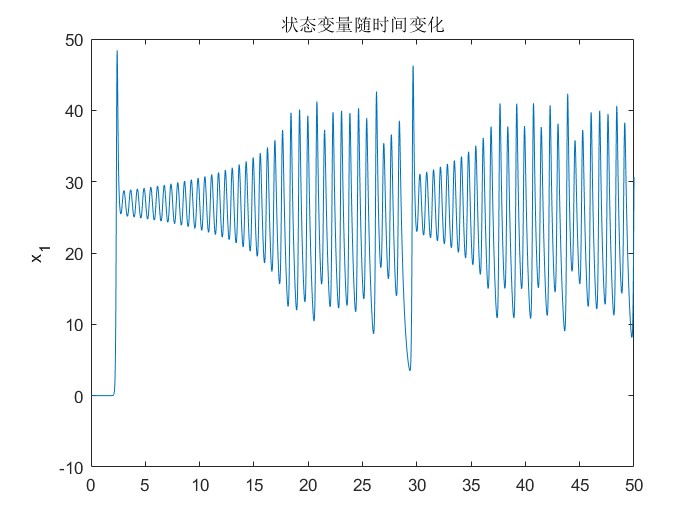
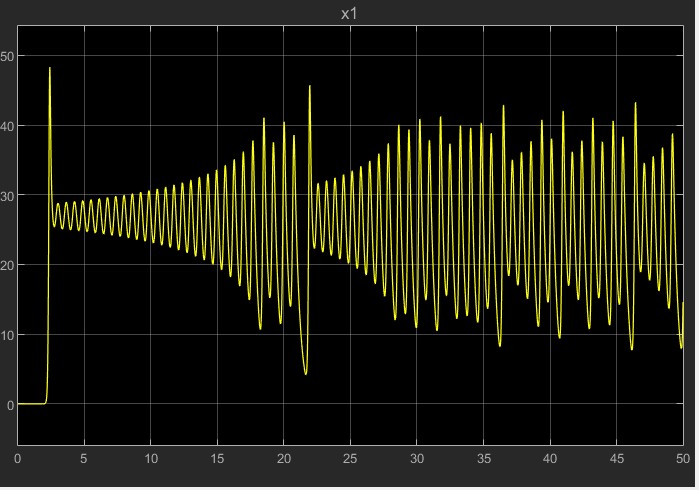


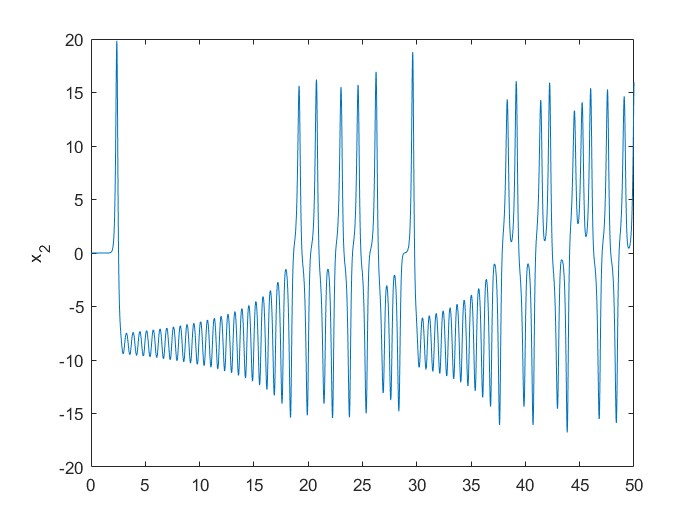
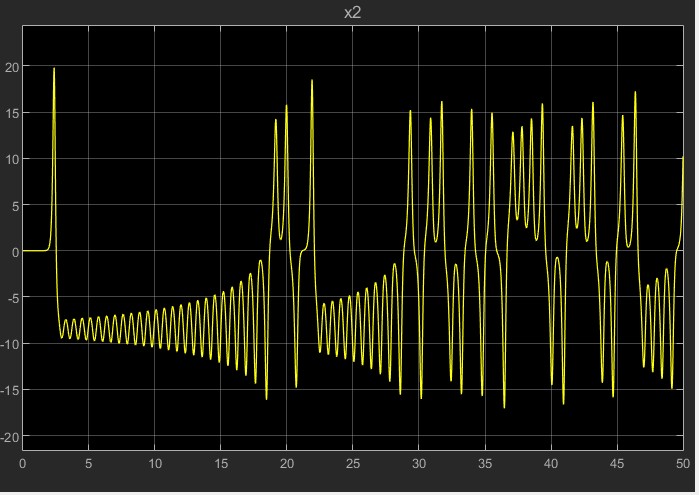
最终得到模型如下图所示：

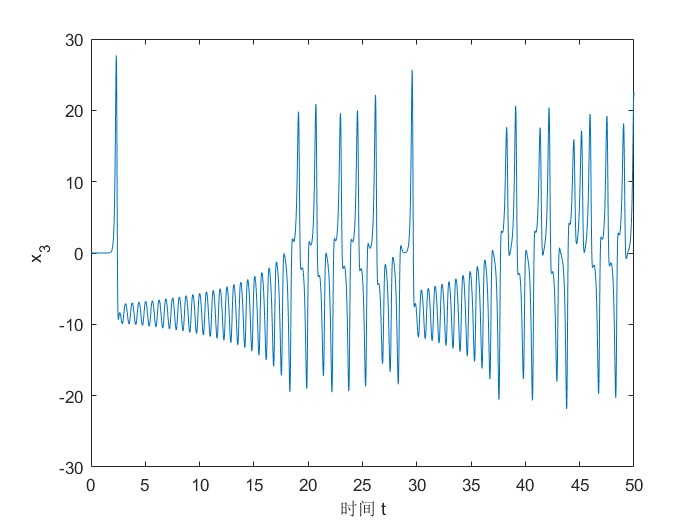


同时运用matlab程序进行求解结果对比，程序源码详见附录。

结果如下图所示（左侧图为SIMULINK仿真结果，右侧图为MATLAB程序运行结果）：





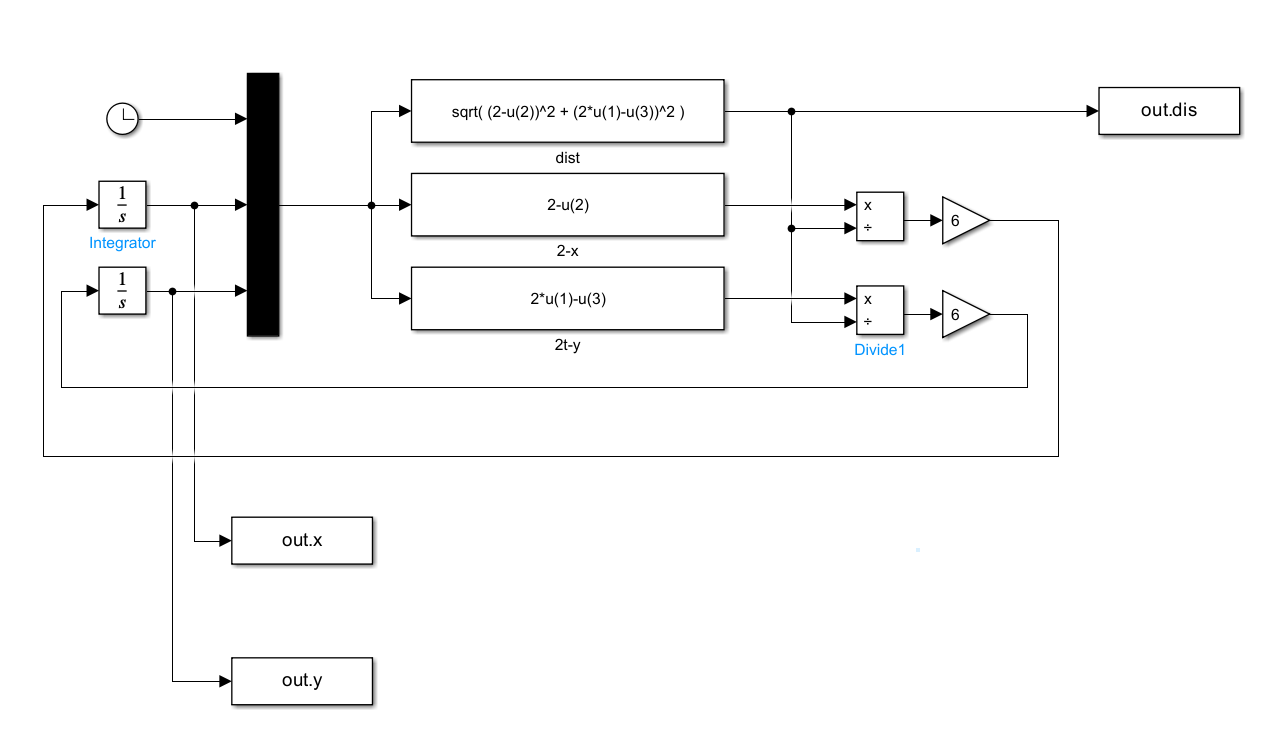


## 应用题七

设位于坐标原点的甲舰向位于x轴上点A(2,0)处的乙舰发射导弹,导弹头始终对准乙舰。

如果乙舰以最大的速度2m/s沿平行于y轴的直线行驶,导弹的速度是6m/s,用MATLAB/SIMULINK求解导弹的运动轨迹,给出导弹击中时,乙舰行驶的时间。

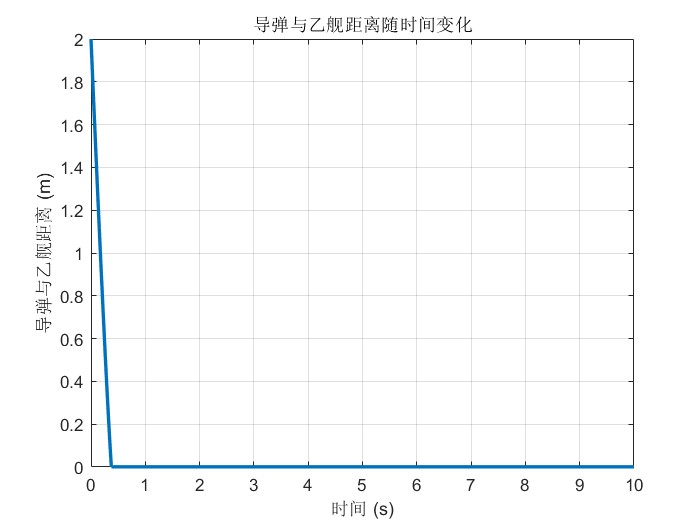
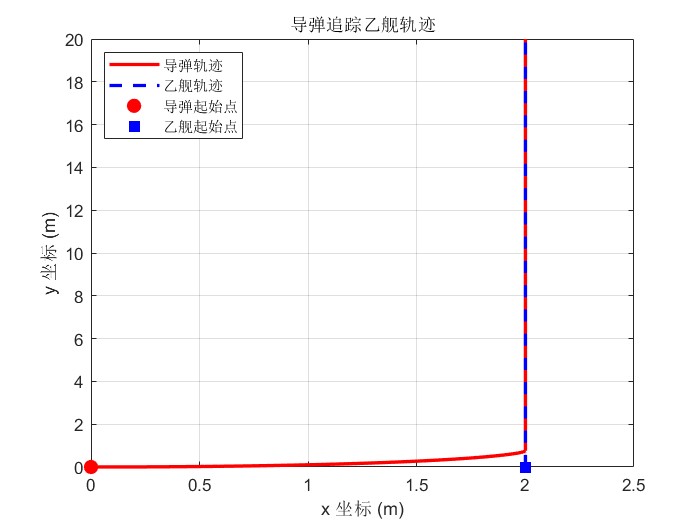
根据题目构建如下系统：



将所求通过TO WORKSPACE传到工作区，从而通过matlab代码进行绘图：

|  |
| --- |
| clc;clear;  simout = sim('y7s.slx');  t = simout.x.Time;  x = simout.x.Data;  y = simout.y.Data;  dis = simout.dis.Data;  %% 舰船  x0\_target = 2; y0\_target = 0; % 乙舰初始位置 (2,0)  dt = 0.001;  t\_max = 10;  t = 0:dt:t\_max;  x\_target = zeros(size(t));  y\_target = zeros(size(t));  v\_target = 2;  x\_target(1) = x0\_target;  y\_target(1) = y0\_target;  for i = 2:length(t)  % 乙舰位置 (沿y轴匀速运动)  x\_target(i) = x0\_target;  y\_target(i) = y0\_target + v\_target \* t(i);  end  % 绘制轨迹  figure;  plot(x, y, 'r-', 'LineWidth', 2); hold on;  plot(x\_target, y\_target, 'b--', 'LineWidth', 2);  plot(0, 0, 'ro', 'MarkerSize', 8, 'MarkerFaceColor', 'r');  plot(x0\_target, y0\_target, 'bs', 'MarkerSize', 8, 'MarkerFaceColor', 'b');  xlabel('x 坐标 (m)');  ylabel('y 坐标 (m)');  title('导弹追踪乙舰轨迹');  legend('导弹轨迹', '乙舰轨迹', '导弹起始点', '乙舰起始点', '击中点', ...  'Location', 'northwest');  grid on;  % 绘制距离随时间变化  figure;  plot(t, dis, 'LineWidth', 2);  xlabel('时间 (s)');  ylabel('导弹与乙舰距离 (m)');  title('导弹与乙舰距离随时间变化');  grid on; |

最终得出结果如下图所示：



# 综合题

## 综合题八

已知飞行器六自由度十二状态方程（无动力）描述如下：







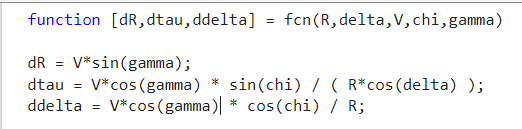
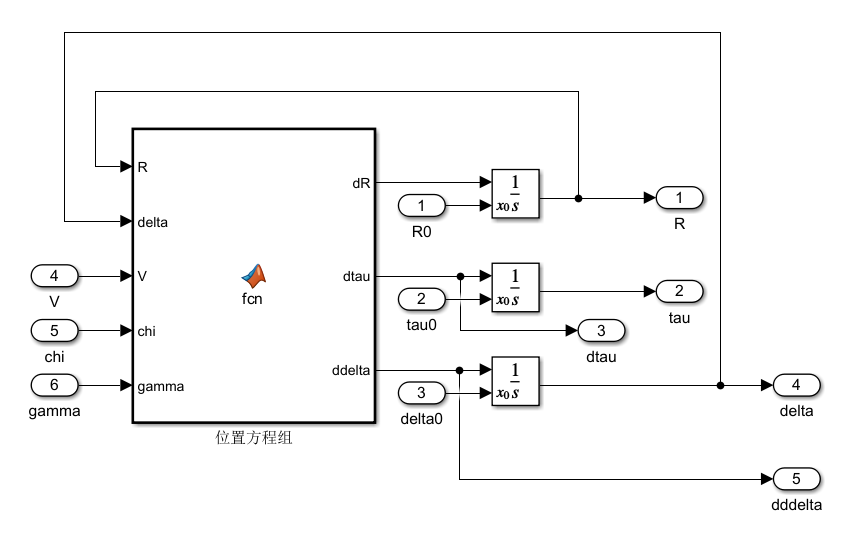
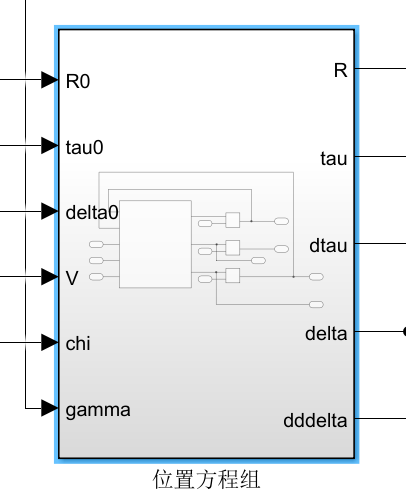


其中 分别为地心到飞行器质心的距离、飞行器所处经度和纬度；分别是飞行速度（设无风）、航迹方位角和航迹倾角；分别为迎角、侧滑角和航迹滚转角；分别为飞行器绕体轴的滚转、俯仰和偏航角速度；是飞行器质量，转动惯量 以及交叉项 （质心与焦点的距离）都是质点的函数。

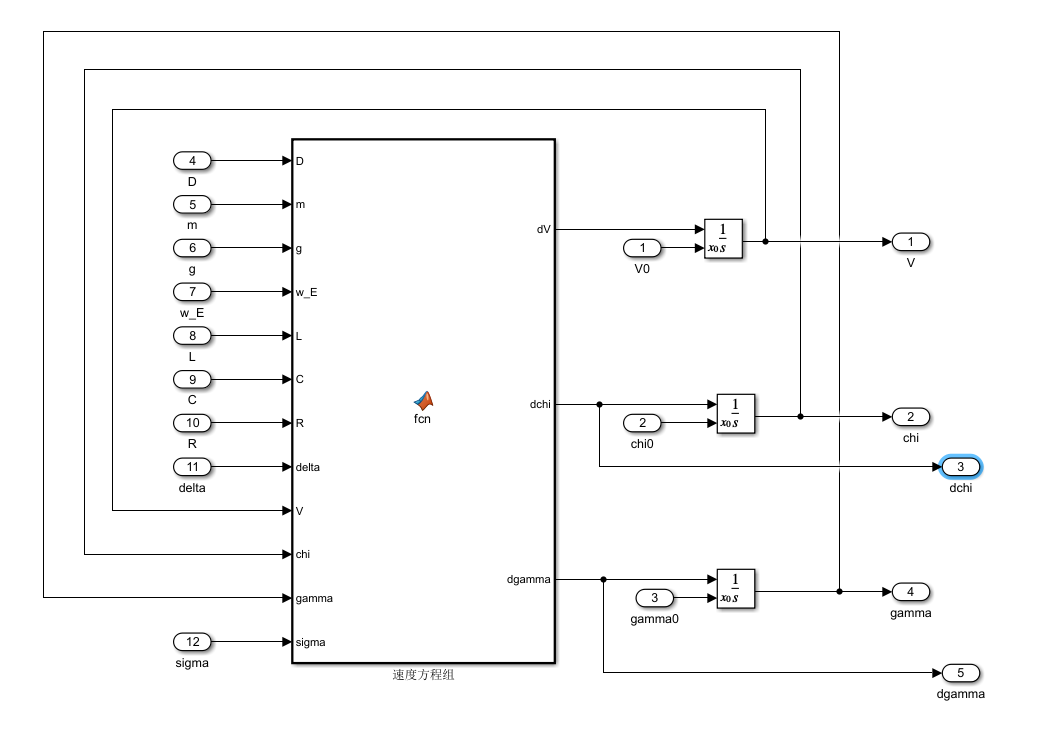
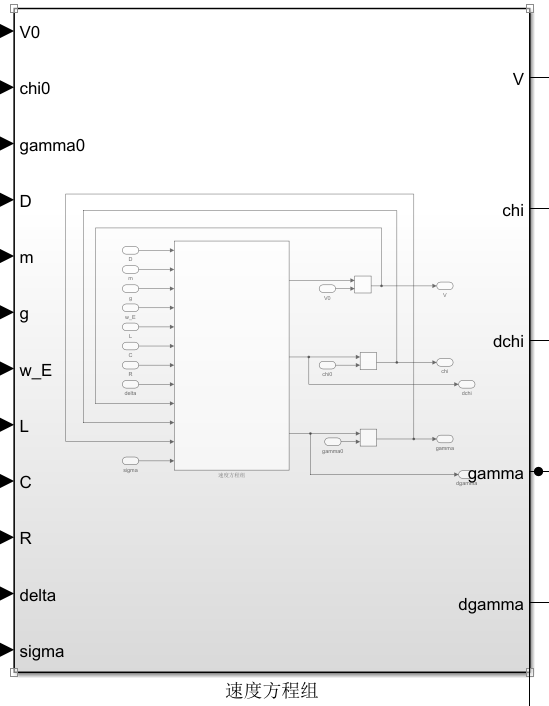
根据某飞行器模型参数，用MATLAB/SIMULINK搭建仿真模型，其中初始值为，给出飞行器模型输出变化的曲线。

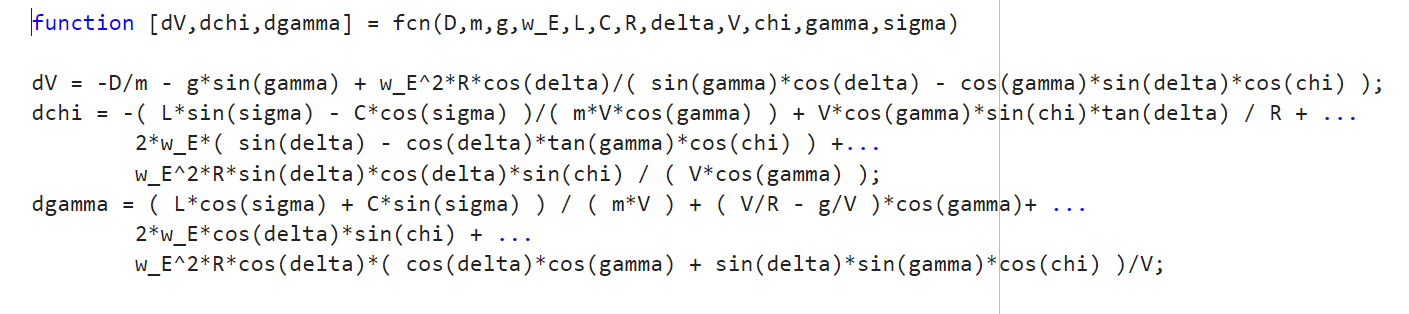
根据题目构建如下系统：

### 位置方程组建模

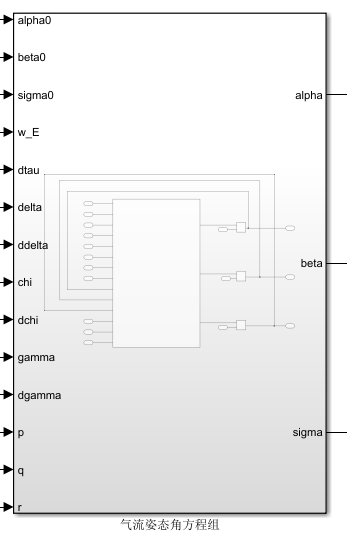
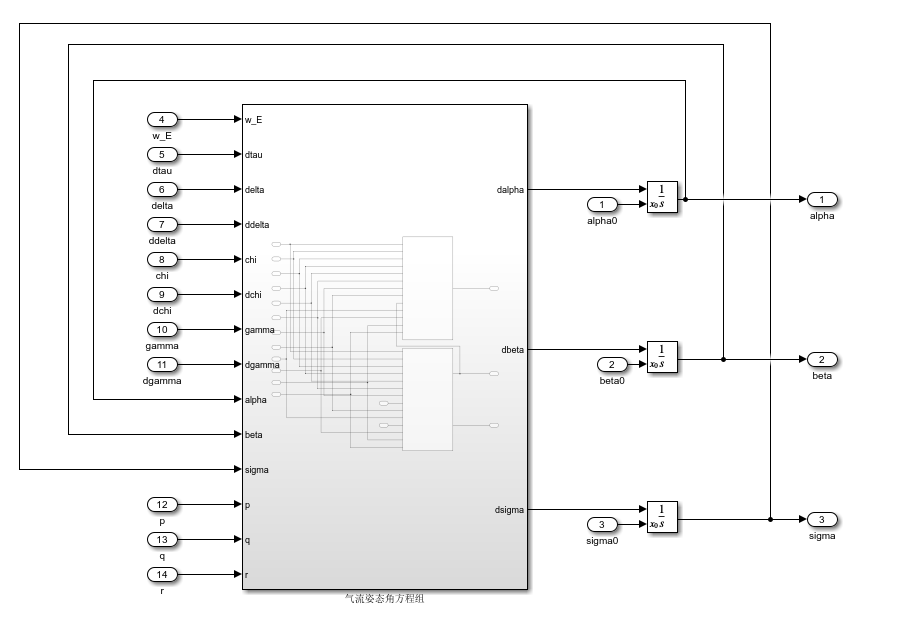


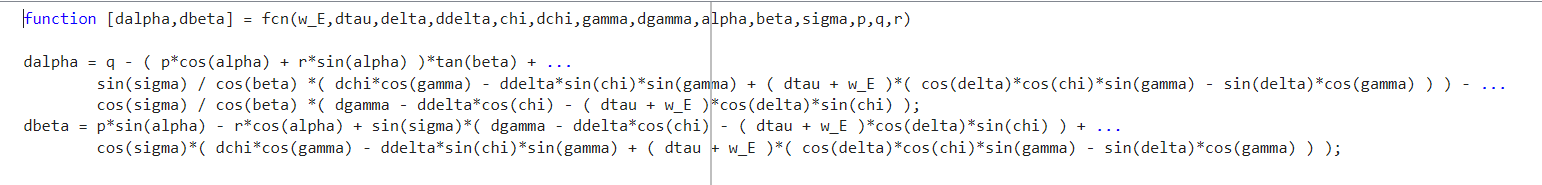
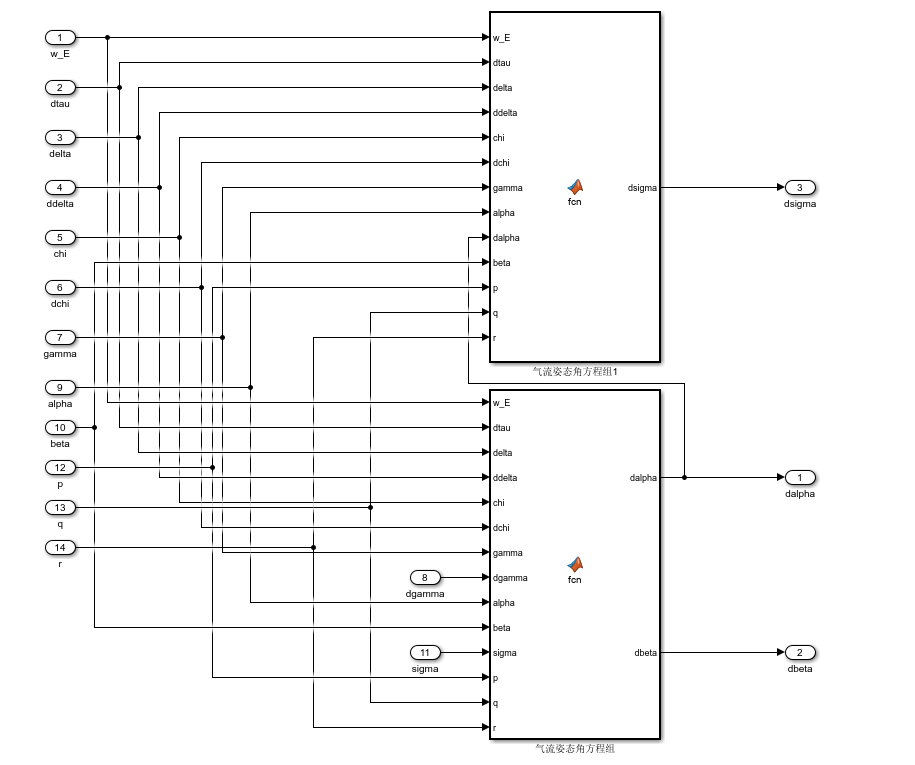
### 速度方程组建模

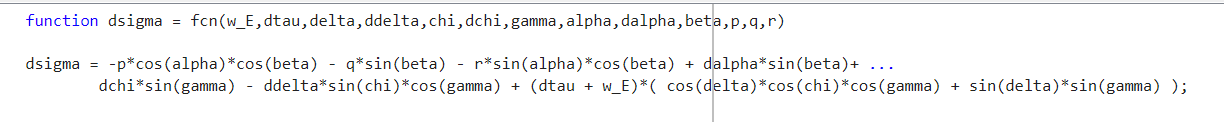




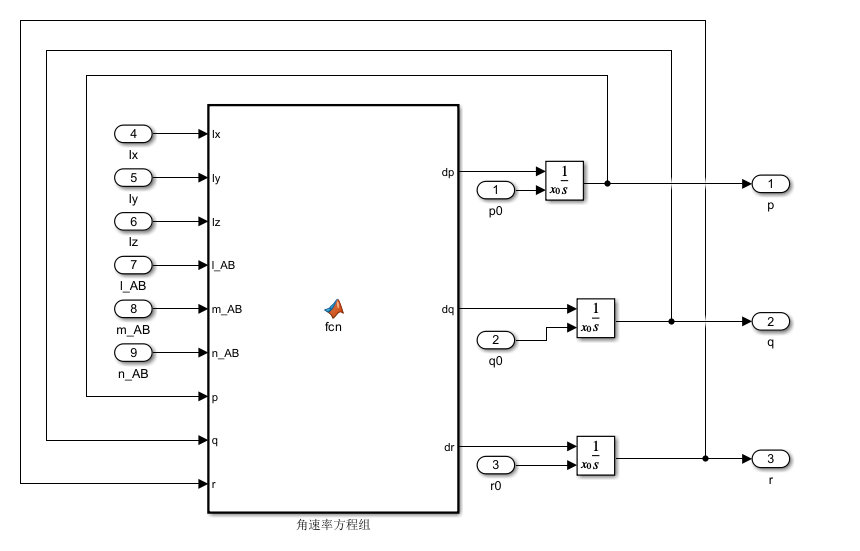
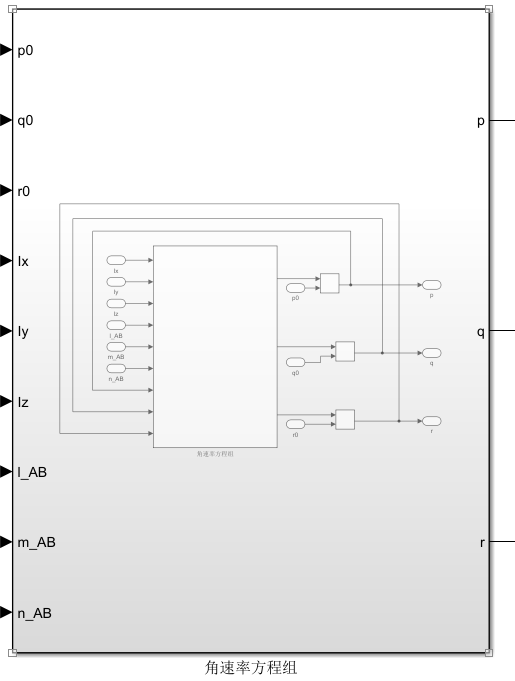
### 气流姿态角方程组

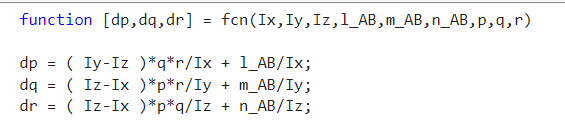
 



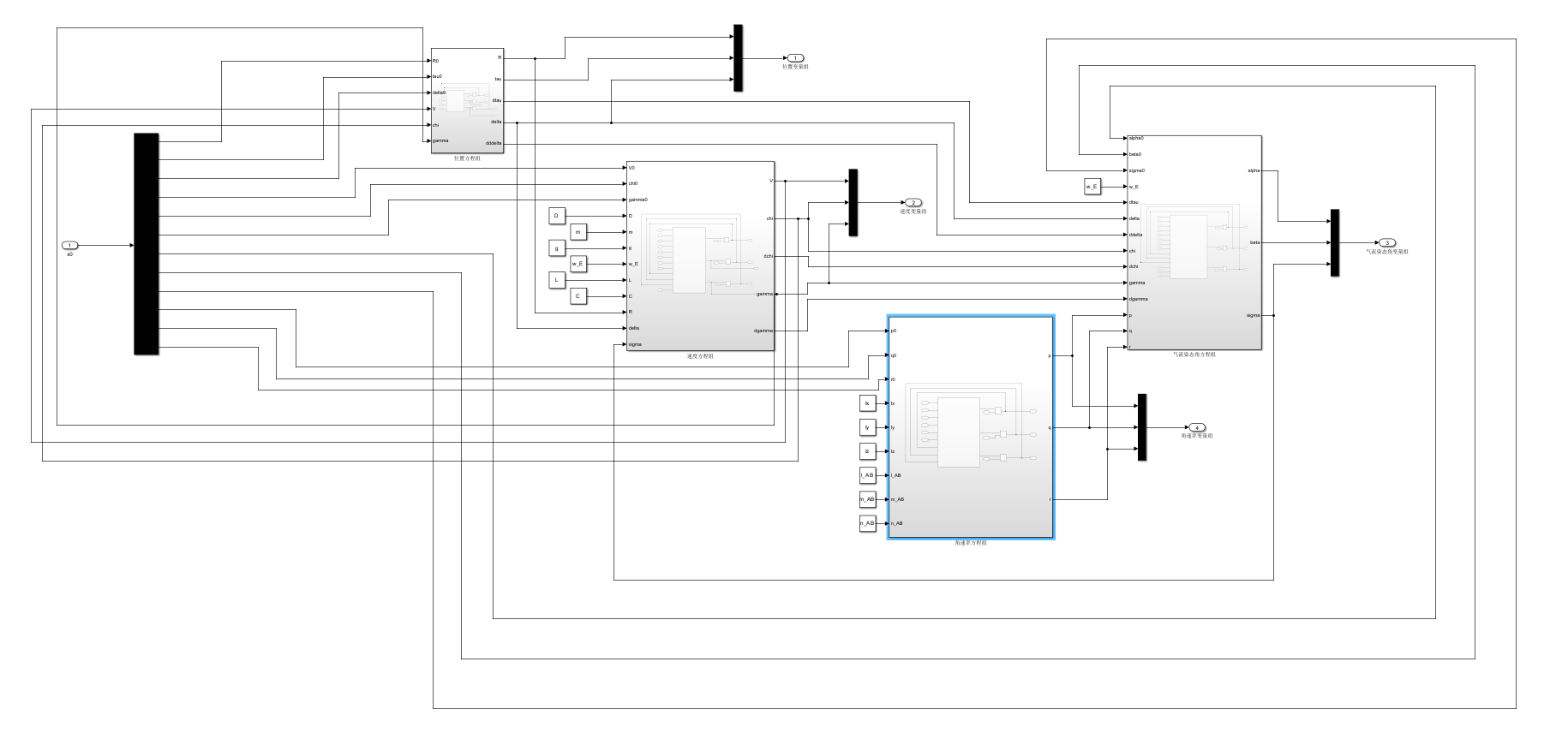


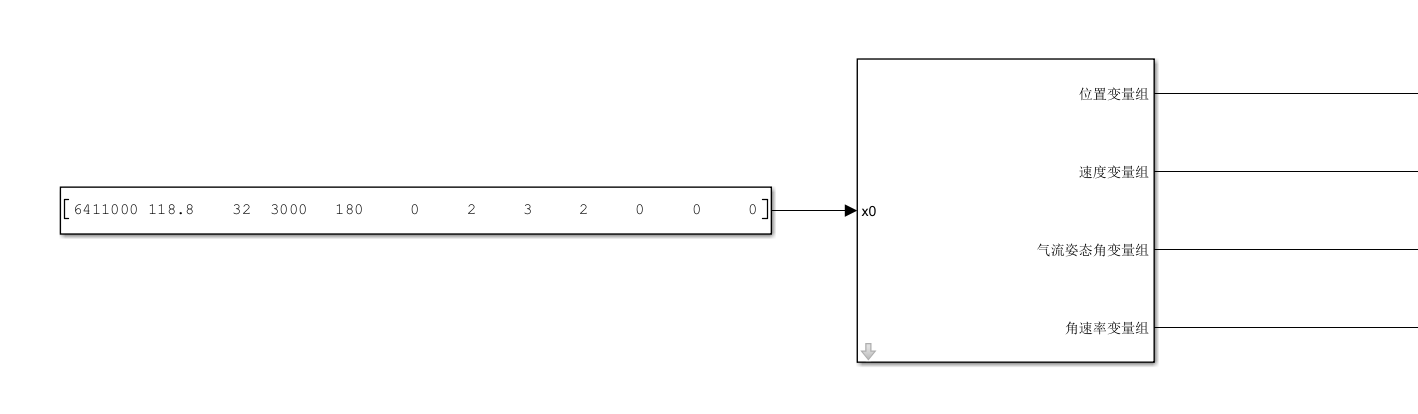
### 角速率方程组

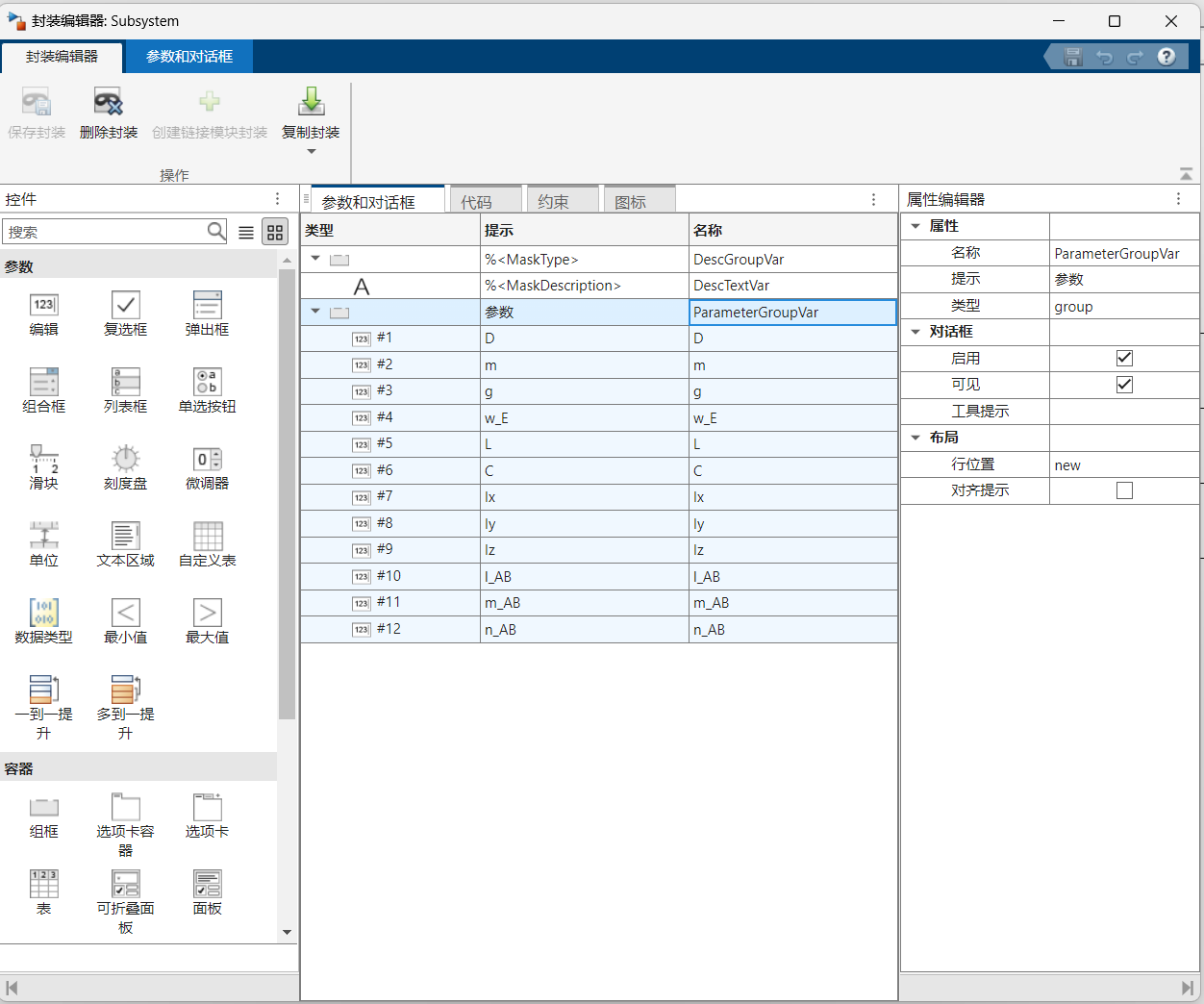




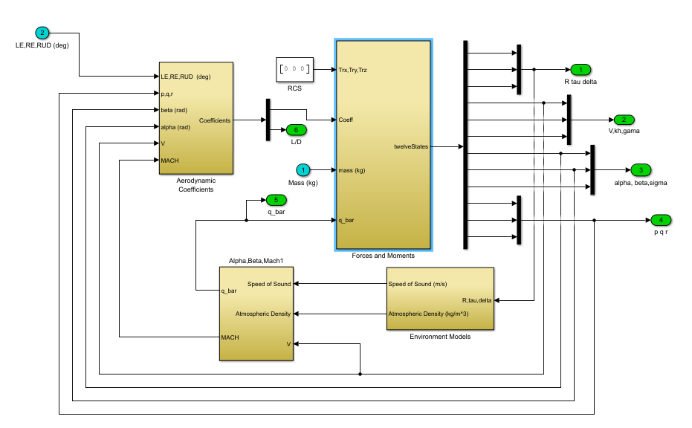
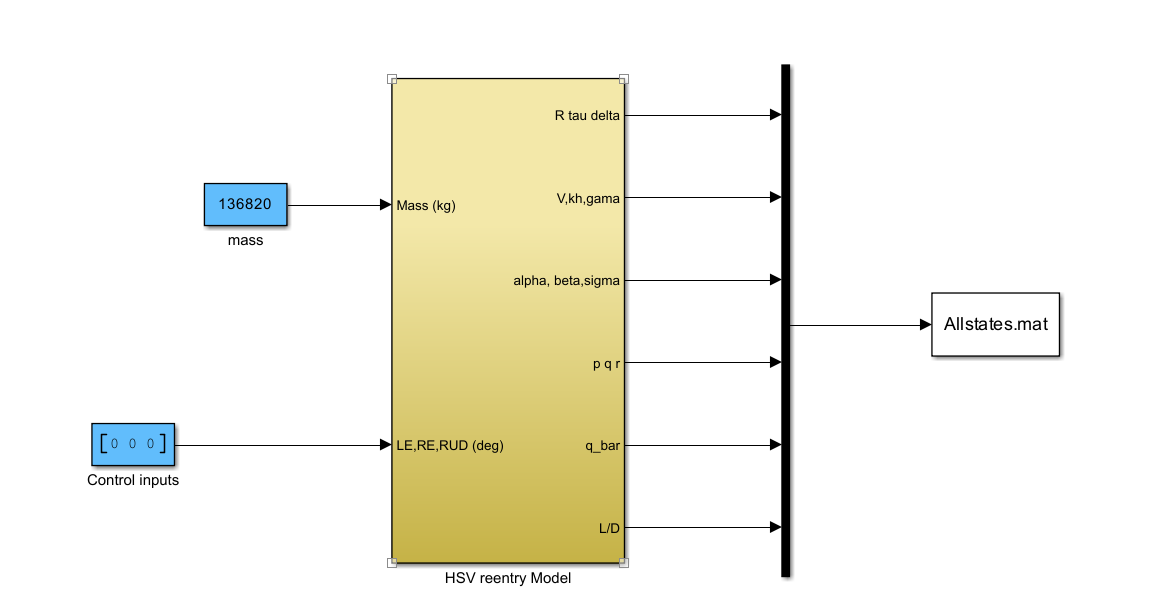
### 模型封装

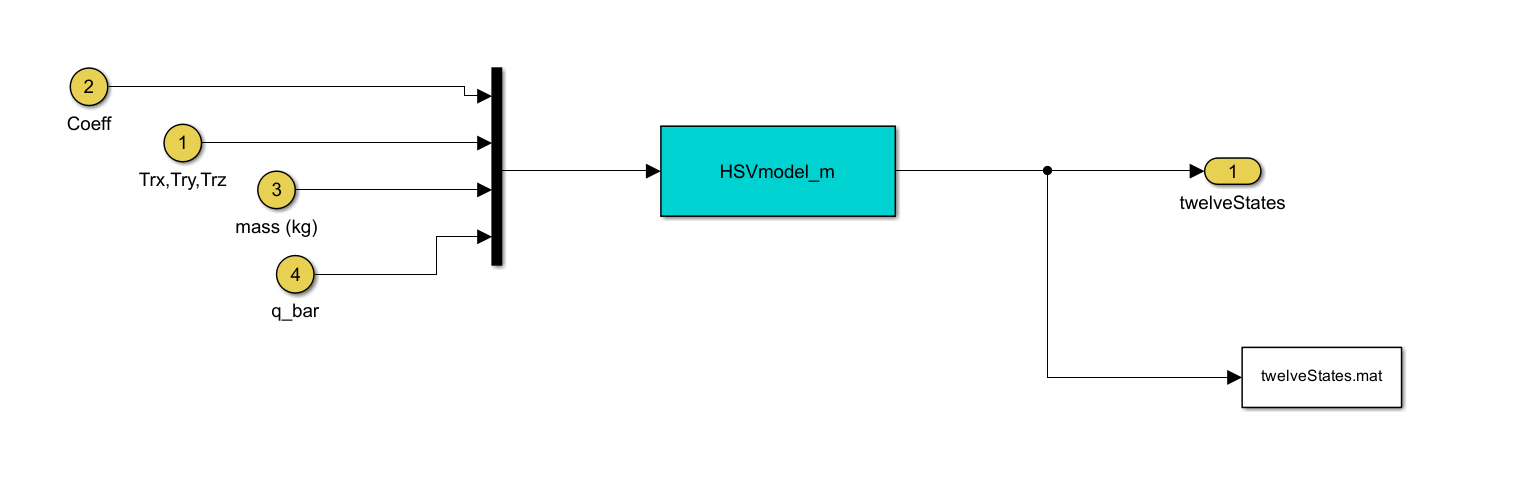






## 真模型





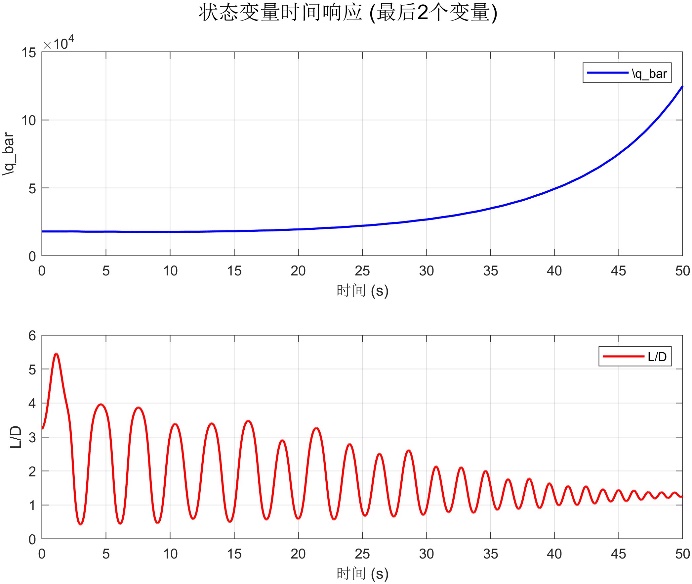
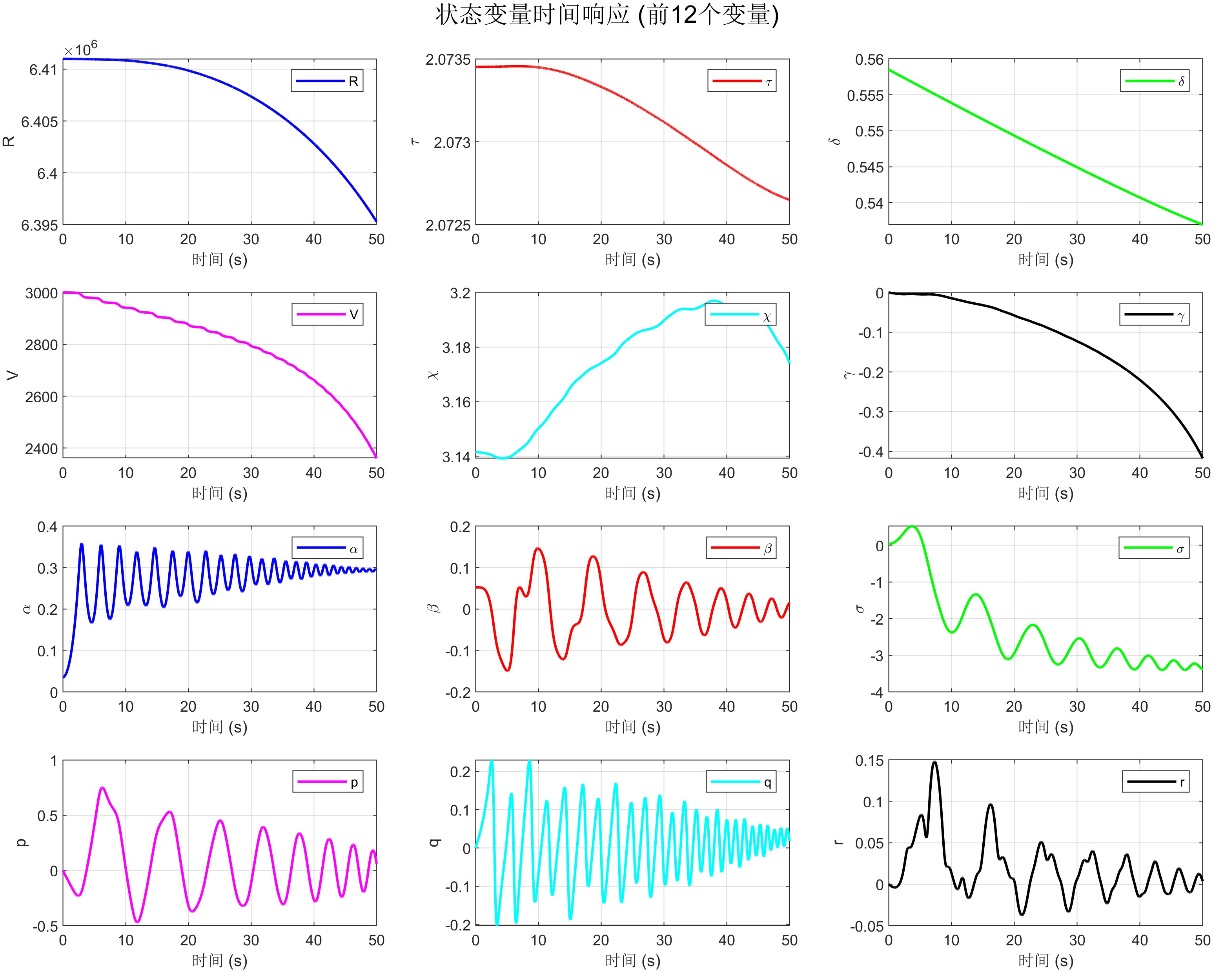
### 代码补全

|  |
| --- |
| % initialize the initial conditions  %  x0 = [6411000;118.8\*d2r;32\*d2r;3000;180\*d2r;0\*d2r;2\*d2r;3\*d2r;2\*d2r;0\*d2r;0\*d2r;0\*d2r]; %设12个状态的初始值  %  % str is always an empty matrix  %  str = [];  %  % initialize the array of sample times  %  ts = [0 0];  % end mdlInitializeSizes  %  %=============================================================================  % mdlDerivatives  % Return the derivatives for the continuous states.  %=============================================================================  %  function sys=mdlDerivatives(t,x,u)  m2ft = 3.28084; %米到英尺  X\_mrc = 124.01/m2ft;  g = 9.65;  S\_ref = 334.73; %参考面积  c\_ref = 24.384; %平均气动弦长  b\_ref = 18.288; %翼展长度  X\_thrust= 60; %主发动机推力中心到前缘距离  X\_rc = 7.5/m2ft/2; %摆动发动机推力作用线与机体纵轴之间的距离  CD = u(1);  CL = u(2);  CC = u(3); %以上三个为阻力，侧力和升力系数  Cll = u(4);  Cm = u(5);  Cn = u(6); %以上三个为气动力矩系数  Trx = u(7);  Try = u(8);  Trz = u(9); %以上三个为发动机产生的推力矢量在机体轴上的分解  M = u(10); %飞行器瞬时质量  q\_bar = u(11); %动压  R = x(1); %地心指向飞行器质心的位置矢量大小  tau = x(2); %经度  delta = x(3); %纬度  V = x(4); %飞行速度  kh = x(5); %航迹方位角  gamma = x(6); %航迹倾斜角  alpha = x(7); %迎角  beta = x(8); %侧滑角  sigma = x(9); %航迹滚转角  p = x(10);  q = x(11);  r = x(12); %以上三个为角速率  Ixx = 1.0e+004 \*(-0.00000000709597\*M^2+ 0.00199093623971\*M -5.94301903021396) ;  Iyy = 1.0e+006 \*(-0.00000000080331\*M^2+ 0.00021974209927\*M -1.69055469296301) ;  Izz = Iyy; %三个转动惯量　 (方程10到12使用)  X\_cg = 0.00000000016469\*M^2 -0.00005571500480\*M+ 7.36721170979675; %质心到mrc的距离  X\_mass = X\_mrc + X\_cg; %质心到前缘的距离  D = q\_bar\*S\_ref\*CD; %阻力 (方程4)  C = q\_bar\*S\_ref\*CC; %侧力 (方程5和6)  L = q\_bar\*S\_ref\*CL; %升力 (方程5和6)  m\_mrc = q\_bar\*c\_ref\*S\_ref\*Cm;  n\_mrc = q\_bar\*b\_ref\*S\_ref\*Cn;  Z = D\*sin(alpha)+L\*cos(alpha);  l\_A = q\_bar\*b\_ref\*S\_ref\*Cll; %滚转力矩 (方程10)  m\_A = m\_mrc + X\_cg\*Z; %俯仰力矩 (方程11)  n\_A = n\_mrc + X\_cg\*C\*cos(beta); %偏航力矩 (方程12)  wcb = 0.000073; %地球自转角速率  %(方程1)  dR = V\*sin(gamma);  %(方程2)  dtau = V\*cos(gamma) \* sin(kh) / ( R\*cos(delta) );  %(方程3)  ddelta = V\*cos(gamma) \* cos(kh) / R; %(方程3)  %(方程4)  dV = -D/M - g\*sin(gamma) + wcb^2\*R\*cos(delta)\*( sin(gamma)\*cos(delta) - cos(gamma)\*sin(delta)\*cos(kh) );  %(方程5)  dkh = -( L\*sin(sigma) - C\*cos(sigma) )/( M\*V\*cos(gamma) ) + V\*cos(gamma)\*sin(kh)\*tan(delta) / R + ...  2\*wcb\*( sin(delta) - cos(delta)\*tan(gamma)\*cos(kh) ) +...  wcb^2\*R\*sin(delta)\*cos(delta)\*sin(kh) / ( V\*cos(gamma) );  %(方程6)  dgamma = ( L\*cos(sigma) + C\*sin(sigma) ) / ( M\*V ) + ( V/R - g/V )\*cos(gamma)+ ...  2\*wcb\*cos(delta)\*sin(kh) + ...  wcb^2\*R\*cos(delta)\*( cos(delta)\*cos(gamma) + sin(delta)\*sin(gamma)\*cos(kh) )/V;  %(方程7)  dalpha = q -tan(beta)\*(p\*cos(alpha)+r\*sin(alpha))+ sin(sigma)/cos(beta)\*...  (dkh\*cos(gamma)- ddelta\*sin(gamma)\*sin(kh)+(dtau+wcb)\*(cos(delta)\*cos(kh)\*sin(gamma)-sin(delta)\*cos(gamma)))...  -cos(sigma)/cos(beta)\*( dgamma-ddelta\*cos(kh)-(dtau+wcb)\*cos(delta)\*sin(kh));  %(方程7)  dbeta = p\*sin(alpha) - r\*cos(alpha) + sin(sigma)\*( dgamma - ddelta\*cos(kh) - ( dtau + wcb )\*cos(delta)\*sin(kh) ) + ...  cos(sigma)\*( dkh\*cos(gamma) - ddelta\*sin(kh)\*sin(gamma) + ( dtau + wcb )\*( cos(delta)\*cos(kh)\*sin(gamma) - sin(delta)\*cos(gamma) ) );  %(方程9)  dsigma = -p\*cos(alpha)\*cos(beta) - q\*sin(beta) - r\*sin(alpha)\*cos(beta) + dalpha\*sin(beta)- ...  dkh\*sin(gamma) - ddelta\*sin(kh)\*cos(gamma) + (dtau + wcb)\*( cos(delta)\*cos(kh)\*cos(gamma) + sin(delta)\*sin(gamma) );  %(方程10)  dp = ( Iyy-Izz )\*q\*r/Ixx + l\_A/Ixx;  %(方程11)  dq = ( Izz-Ixx )\*p\*r/Iyy + m\_A/Iyy;  %(方程12)  dr = ( Ixx-Iyy )\*p\*q/Izz + n\_A/Izz;  %%%%%%%%%%%%%%%%%%%%------以上为再入飞行6自由度12状态方程（无动力）-------%%%%%%%%%%%%%%%% |

### 绘图程序

|  |
| --- |
| clc;clear;  simout = load("Allstates.mat");  t = simout.Allstates(1,:)';  R = simout.Allstates(2,:)';  tau = simout.Allstates(3,:)';  delta = simout.Allstates(4,:)';  V = simout.Allstates(5,:)';  chi = simout.Allstates(6,:)';  gamma = simout.Allstates(7,:)';  alpha = simout.Allstates(8,:)';  beta = simout.Allstates(9,:)';  sigma = simout.Allstates(10,:)';  p = simout.Allstates(11,:)';  q = simout.Allstates(12,:)';  r = simout.Allstates(13,:)';  q\_bar = simout.Allstates(14,:)';  LD = simout.Allstates(15,:)';  % 创建第一个图形窗口 - 4x3子图（前12个变量）  figure('Position', [100, 100, 1200, 900]);  % 第1行  subplot(4,3,1);  plot(t, R, 'b-', 'LineWidth', 1.5);  xlabel('时间 (s)'); ylabel('R');  grid on;  legend('R');  subplot(4,3,2);  plot(t, tau, 'r-', 'LineWidth', 1.5);  xlabel('时间 (s)'); ylabel('\tau');  grid on;  legend('\tau');  subplot(4,3,3);  plot(t, delta, 'g-', 'LineWidth', 1.5);  xlabel('时间 (s)'); ylabel('\delta');  grid on;  legend('\delta');  % 第2行  subplot(4,3,4);  plot(t, V, 'm-', 'LineWidth', 1.5);  xlabel('时间 (s)'); ylabel('V');  grid on;  legend('V');  subplot(4,3,5);  plot(t, chi, 'c-', 'LineWidth', 1.5);  xlabel('时间 (s)'); ylabel('\chi');  grid on;  legend('\chi');  subplot(4,3,6);  plot(t, gamma, 'k-', 'LineWidth', 1.5);  xlabel('时间 (s)'); ylabel('\gamma');  grid on;  legend('\gamma');  % 第3行  subplot(4,3,7);  plot(t, alpha, 'b-', 'LineWidth', 1.5);  xlabel('时间 (s)'); ylabel('\alpha');  grid on;  legend('\alpha');  subplot(4,3,8);  plot(t, beta, 'r-', 'LineWidth', 1.5);  xlabel('时间 (s)'); ylabel('\beta');  grid on;  legend('\beta');  subplot(4,3,9);  plot(t, sigma, 'g-', 'LineWidth', 1.5);  xlabel('时间 (s)'); ylabel('\sigma');  grid on;  legend('\sigma');  % 第4行  subplot(4,3,10);  plot(t, p, 'm-', 'LineWidth', 1.5);  xlabel('时间 (s)'); ylabel('p');  grid on;  legend('p');  subplot(4,3,11);  plot(t, q, 'c-', 'LineWidth', 1.5);  xlabel('时间 (s)'); ylabel('q');  grid on;  legend('q');  subplot(4,3,12);  plot(t, r, 'k-', 'LineWidth', 1.5);  xlabel('时间 (s)'); ylabel('r');  grid on;  legend('r');  sgtitle('状态变量时间响应 (前12个变量)');  % 创建第二个图形窗口 - 2x1子图（最后2个变量）  figure('Position', [100, 100, 800, 600]);  subplot(2,1,1);  plot(t, q\_bar, 'b-', 'LineWidth', 1.5);  xlabel('时间 (s)'); ylabel('\q\_bar');  grid on;  legend('\q\_bar');  subplot(2,1,2);  plot(t, LD, 'r-', 'LineWidth', 1.5);  xlabel('时间 (s)'); ylabel('L/D');  grid on;  legend('L/D');  sgtitle('状态变量时间响应 (最后2个变量)');  % 调整所有图形的显示  disp('图形绘制完成！');  exportgraphics(figure(1), 'E:\Programing\MATLAB\feixingqi\shangji2\1.jpg', 'Resolution', 300);  exportgraphics(figure(2), 'E:\Programing\MATLAB\feixingqi\shangji2\2.jpg', 'Resolution', 300); |

### 结果



## 综合题九

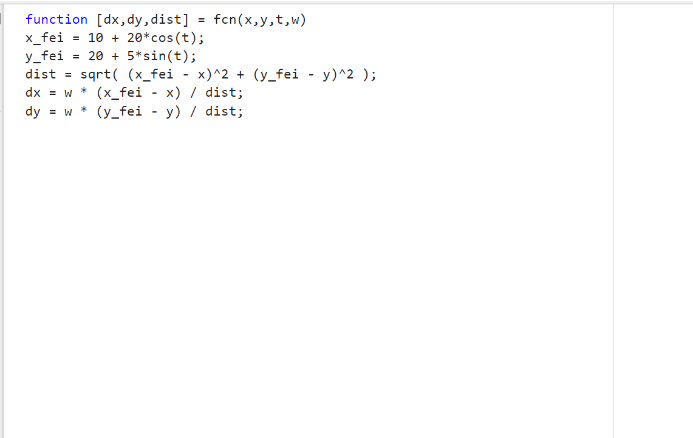
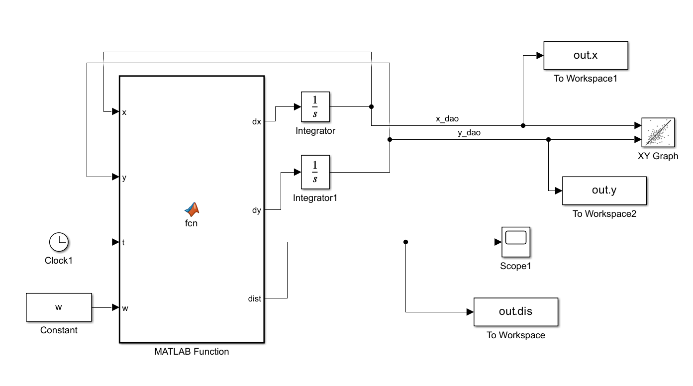
用MATLAB/SIMULINK进行数学仿真，求两种情形下导弹能够摧毁飞行器所需的最小速率w，并给出两种情形的对比分析结果：

情形一：某飞行器在平面上以一定速度运动，其运动椭圆轨迹方程为： ，且，突然有一枚导弹发现它，这枚导弹从原点出发，以恒定速率w去攻击该飞行器，导弹的运行方向始终指向该飞行器。

情形二：某飞行器在平面上以单位速率运动，其运动椭圆轨迹方程为： 。突然有一枚导弹发现它，从原点出发，以恒定速率w去攻击该飞行器，导弹的运行方向始终指向该飞行器。

### 情形一

不难发现，飞行器以固定角速度运行但飞机速度不恒定，则对情形一进行建模，模型如下：



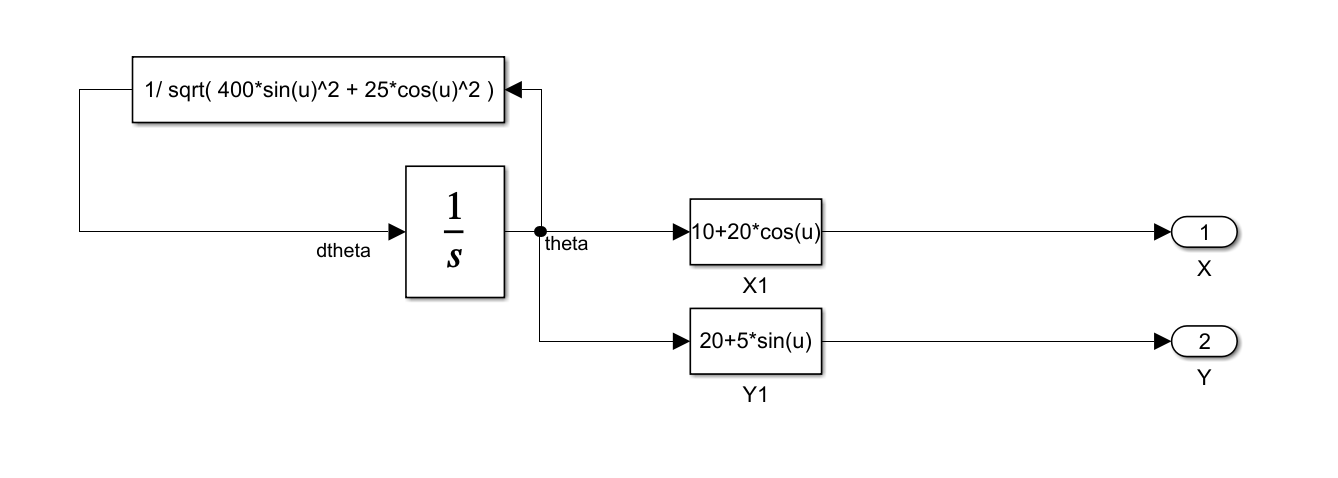
### 情形二

不难发现，飞行器以固定速度运行，则有

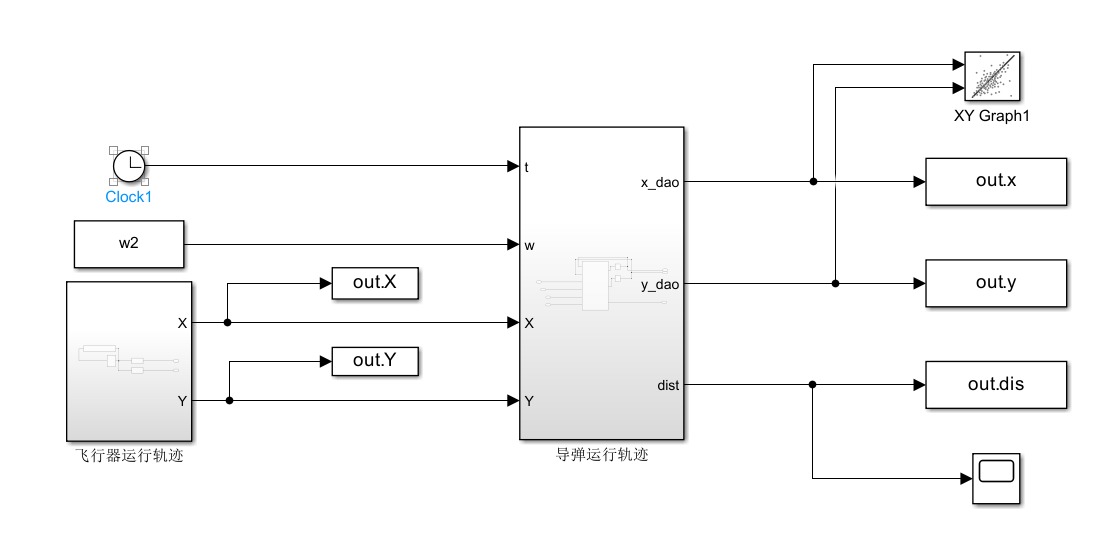


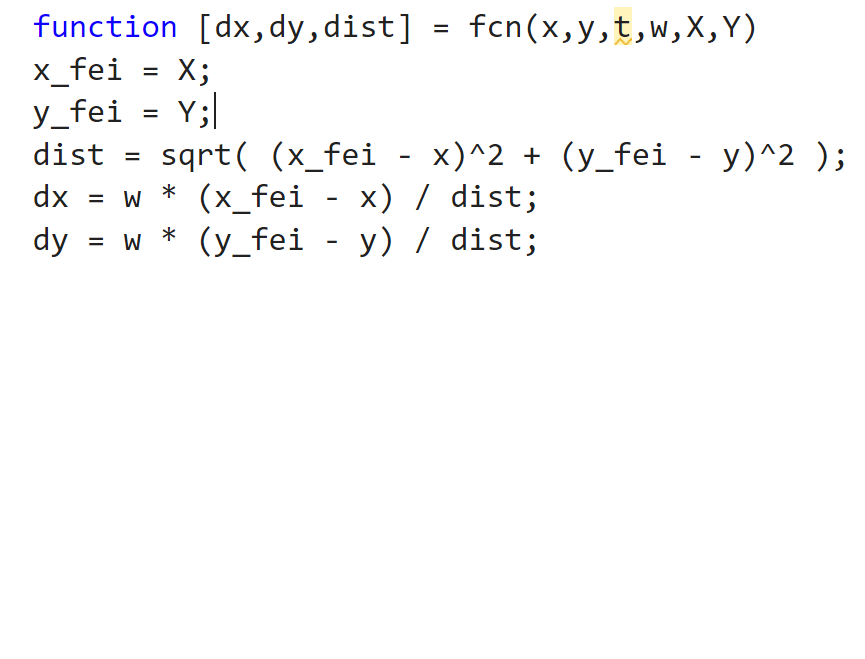
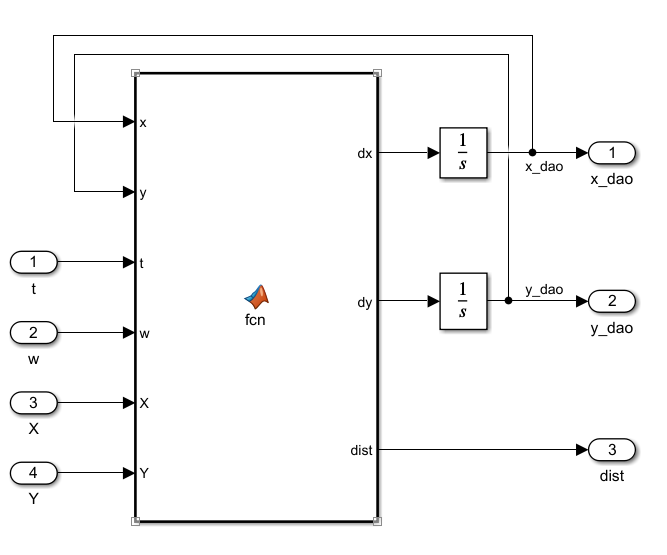


在求解导弹运行轨迹前，应先对飞行器运行轨迹进行建模，如下所示：



最终结合情形1模型建立如下模型



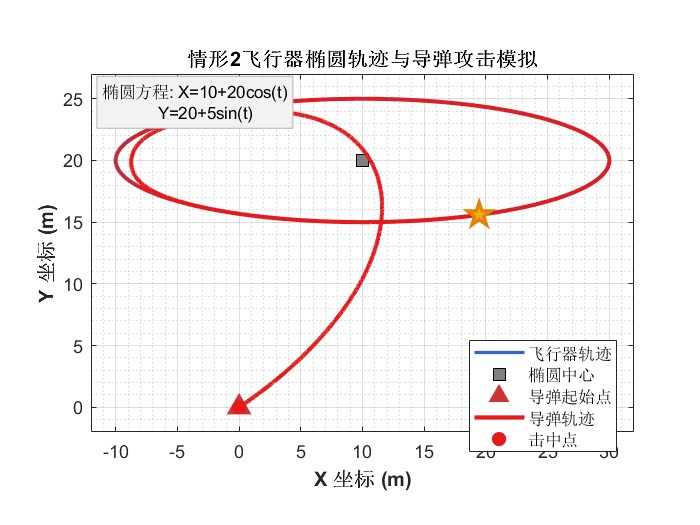
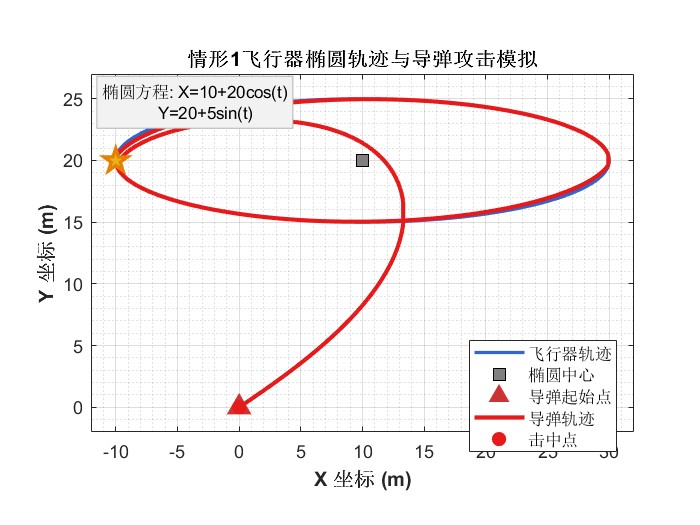


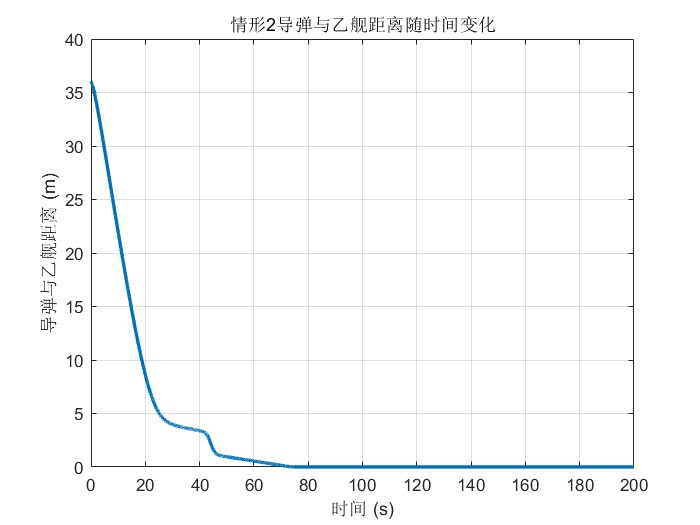
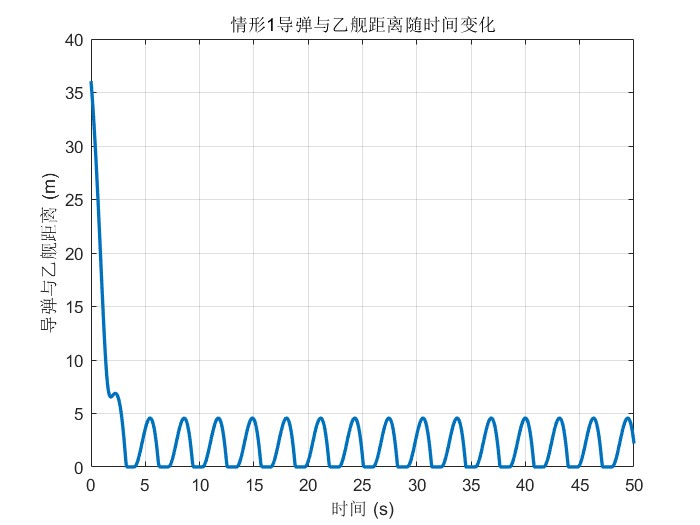
### 情形一和二最小速度求解及结果对比

此处采用与应用题五一样的方法对最小速度进行求解，MATLAB代码如下所示：

|  |
| --- |
| clc;clear;close;  %% 寻找情形1最小速度  for w = 15:0.01:18  simout=sim("tuoyuan.slx");  dis=simout.dis.data;  if(min(dis)<1e-5)  break;  end  end  disp(['模型1导弹最小速度',num2str(w)]);  %% 绘制情形1  % 参数设置  t = simout.dis.Time;  X\_ellipse = 10 + 20 \* cos(t);  Y\_ellipse = 20 + 5 \* sin(t);  % 绘制椭圆轨迹  figure;  plot(X\_ellipse, Y\_ellipse, 'b-', 'LineWidth', 2, 'Color', [0.2 0.4 0.8]);  hold on;  grid on;  % 标记重要点  plot(10, 20, 'ks', 'MarkerSize', 10, 'MarkerFaceColor', [0.5 0.5 0.5]); % 椭圆中心  plot(0, 0, '^', 'MarkerSize', 12, 'MarkerFaceColor', [0.8 0.2 0.2], ...  'Color', [0.8 0.2 0.2]); % 导弹起始点  % 绘制导弹轨迹  x = simout.x.Data;  y = simout.y.Data;  plot(x, y, 'r-', 'LineWidth', 2.5, 'Color', [0.9 0.1 0.1]);  % 标记导弹起点（如果与上面的点重复，可以省略）  plot(0, 0, 'o', 'MarkerSize', 8, 'MarkerFaceColor', [0.9 0.1 0.1], ...  'Color', [0.9 0.1 0.1]);  % 标记击中点（假设击中时距离最小）  [~, hit\_idx] = min(simout.dis.Data);  hit\_x = x(hit\_idx);  hit\_y = y(hit\_idx);  plot(hit\_x, hit\_y, 'pentagram', 'MarkerSize', 15, 'MarkerFaceColor', [0.9 0.7 0.1], ...  'Color', [0.9 0.5 0], 'LineWidth', 2);  % 坐标轴和标题  xlabel('X 坐标 (m)', 'FontSize', 12, 'FontWeight', 'bold');  ylabel('Y 坐标 (m)', 'FontSize', 12, 'FontWeight', 'bold');  title('飞行器椭圆轨迹与导弹攻击模拟', 'FontSize', 14, 'FontWeight', 'bold');  % 图例  legend('飞行器轨迹', '椭圆中心', '导弹起始点', '导弹轨迹', '击中点', ...  'Location', 'best', 'FontSize', 10);  % 图形设置  axis equal;  grid on;  grid minor;  set(gca, 'FontSize', 11);  % 设置坐标轴范围，确保所有轨迹可见  x\_min = min([min(X\_ellipse), min(x), 0]) - 2;  x\_max = max([max(X\_ellipse), max(x), 0]) + 2;  y\_min = min([min(Y\_ellipse), min(y), 0]) - 2;  y\_max = max([max(Y\_ellipse), max(y), 0]) + 2;  axis([x\_min x\_max y\_min y\_max]);  % 添加文本框显示关键信息  text(0.02, 0.98, sprintf('椭圆方程: X=10+20cos(t)\n Y=20+5sin(t)'), ...  'Units', 'normalized', 'VerticalAlignment', 'top', ...  'FontSize', 10, 'BackgroundColor', [0.95 0.95 0.95], ...  'EdgeColor', [0.7 0.7 0.7]);  % 绘制距离随时间变化  figure;  plot(t, dis, 'LineWidth', 2);  xlabel('时间 (s)');  ylabel('导弹与乙舰距离 (m)');  title('导弹与乙舰距离随时间变化');  grid on;  %% 情况2  for w2 = 0:0.01:10  simout2=sim("qx2.slx");  dis2=simout2.dis.data;  % disp(num2str(w2))  if(min(dis2)<1e-2)  break;  end  end  disp(['模型2导弹最小速度',num2str(w2)]);  % 参数设置  t = simout2.dis.Time;  X2 = simout2.X.Data;  Y2 = simout2.Y.Data;  x2 = simout2.x.Data;  y2 = simout2.y.Data;  % 绘制椭圆轨迹  figure;  plot(X2, Y2, 'b-', 'LineWidth', 2, 'Color', [0.2 0.4 0.8]);  hold on;  grid on;  % 标记重要点  plot(10, 20, 'ks', 'MarkerSize', 10, 'MarkerFaceColor', [0.5 0.5 0.5]); % 椭圆中心  plot(0, 0, '^', 'MarkerSize', 12, 'MarkerFaceColor', [0.8 0.2 0.2], ...  'Color', [0.8 0.2 0.2]); % 导弹起始点  % 绘制导弹轨迹  plot(x2, y2, 'r-', 'LineWidth', 2.5, 'Color', [0.9 0.1 0.1]);  % 标记导弹起点（如果与上面的点重复，可以省略）  plot(0, 0, 'o', 'MarkerSize', 8, 'MarkerFaceColor', [0.9 0.1 0.1], ...  'Color', [0.9 0.1 0.1]);  % 标记击中点（假设击中时距离最小）  [~, hit\_idx2] = min(simout2.dis.Data);  hit\_x2 = x2(hit\_idx2);  hit\_y2 = y2(hit\_idx2);  plot(hit\_x2, hit\_y2, 'pentagram', 'MarkerSize', 15, 'MarkerFaceColor', [0.9 0.7 0.1], ...  'Color', [0.9 0.5 0], 'LineWidth', 2);  % 坐标轴和标题  xlabel('X 坐标 (m)', 'FontSize', 12, 'FontWeight', 'bold');  ylabel('Y 坐标 (m)', 'FontSize', 12, 'FontWeight', 'bold');  title('情形2飞行器椭圆轨迹与导弹攻击模拟', 'FontSize', 14, 'FontWeight', 'bold');  % 图例  legend('飞行器轨迹', '椭圆中心', '导弹起始点', '导弹轨迹', '击中点', ...  'Location', 'best', 'FontSize', 10);  % 图形设置  axis equal;  grid on;  grid minor;  set(gca, 'FontSize', 11);  % 设置坐标轴范围，确保所有轨迹可见  x\_min = min([min(X2), min(x2), 0]) - 2;  x\_max = max([max(X2), max(x2), 0]) + 2;  y\_min = min([min(Y2), min(y2), 0]) - 2;  y\_max = max([max(Y2), max(y2), 0]) + 2;  axis([x\_min x\_max y\_min y\_max]);  % 添加文本框显示关键信息  text(0.02, 0.98, sprintf('椭圆方程: X=10+20cos(t)\n Y=20+5sin(t)'), ...  'Units', 'normalized', 'VerticalAlignment', 'top', ...  'FontSize', 10, 'BackgroundColor', [0.95 0.95 0.95], ...  'EdgeColor', [0.7 0.7 0.7]);  % 绘制距离随时间变化  figure;  plot(t, dis2, 'LineWidth', 2);  xlabel('时间 (s)');  ylabel('导弹与乙舰距离 (m)');  title('情形2导弹与乙舰距离随时间变化');  grid on; |

最终求解结果为模型1导弹最小速度15.25，模型2导弹最小速度1.04，结果如下所示：

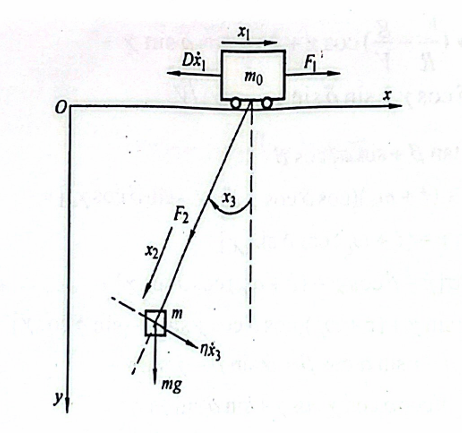




相比较情形2，情形1的飞行器速度较快，当导弹速度较低时很难追上，而情形2的飞行器速度仅有1m/s，所需要的最小速度更小。

## 综合题十

龙门吊车作为一种运载工具,广泛的用于航空航天装备的装卸与运输作业。龙门吊车利用绳索一类的柔性体代替刚体工作,以使得吊车的结构轻便,工作效率高。



另一方面,采用龙门吊车也带来一些负面影响,吊车的摆动问题一直是困扰提高吊车装运效率的难题。已知龙门吊车的精确模型是

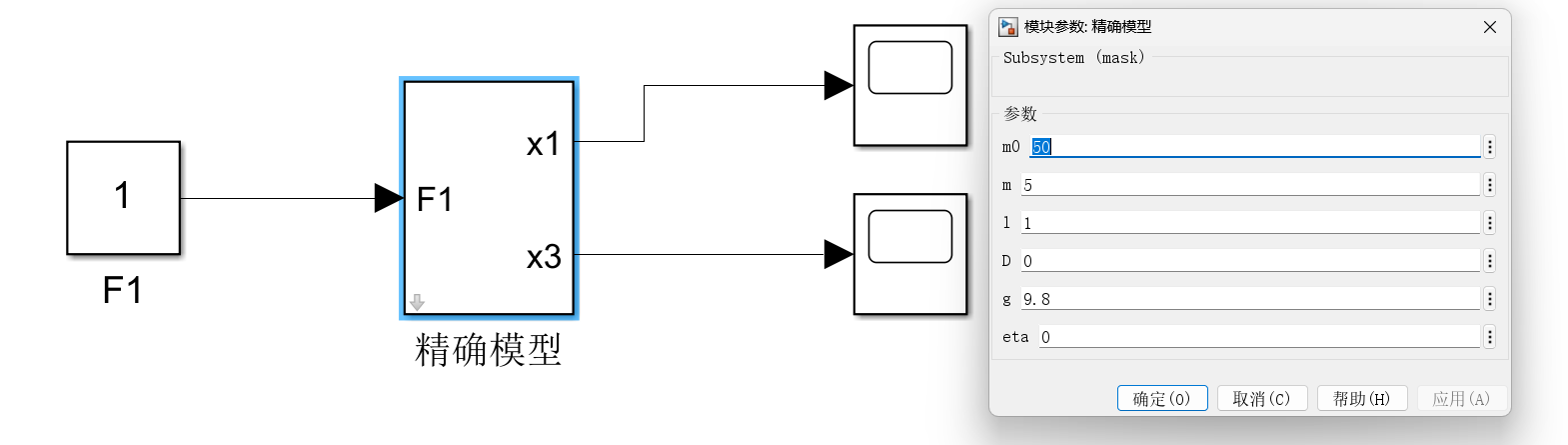


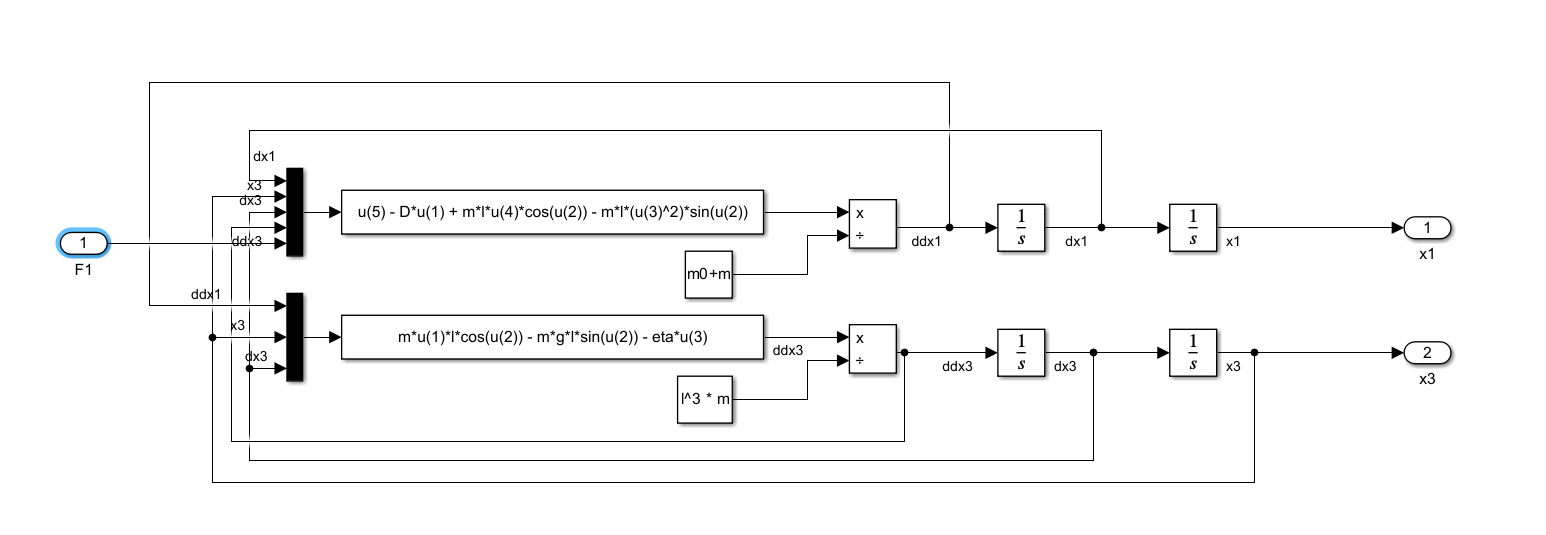
其中小车水平拉力F1的作用下在水平轨道上运动,和分别是小车位移和重物摆角,小车的质量为,重物的质量为,绳索的长度为,小车与水平轨道的摩擦阻尼系数为,重物摆动时的阻尼系数为,其他扰动可忽略。

选取,采用MATLAB/SIMULINK进行仿真模型的构建,对比龙门吊车精确模型和简化模型之间开环响应和闭环响应规律的一致性。

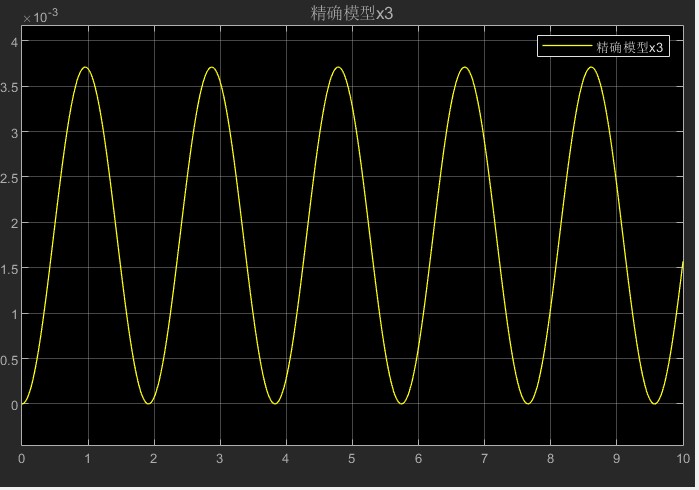
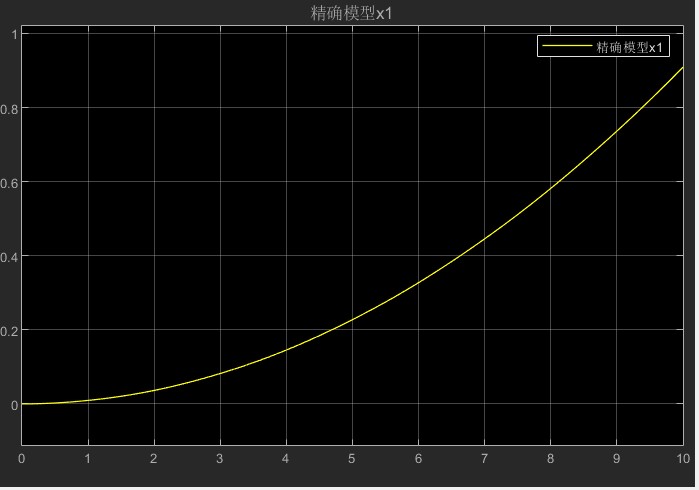
### 精确模型建模

根据题目构建如下精确系统，并将题目中的常量作为未知量进行子系统封装：





最终结果如下所示：



### 简易模型建模

实际吊车运动过程中摆动角较小（不超过十度），且平衡位置为，可将精确模型在处线性化，此时，则精确模型可以化简为



进一步化简可得到：



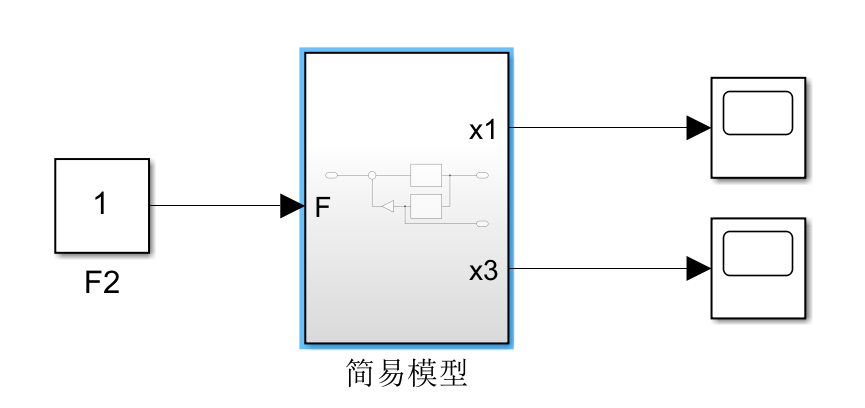
对上式进行拉普拉斯变换便可得到简易模型：

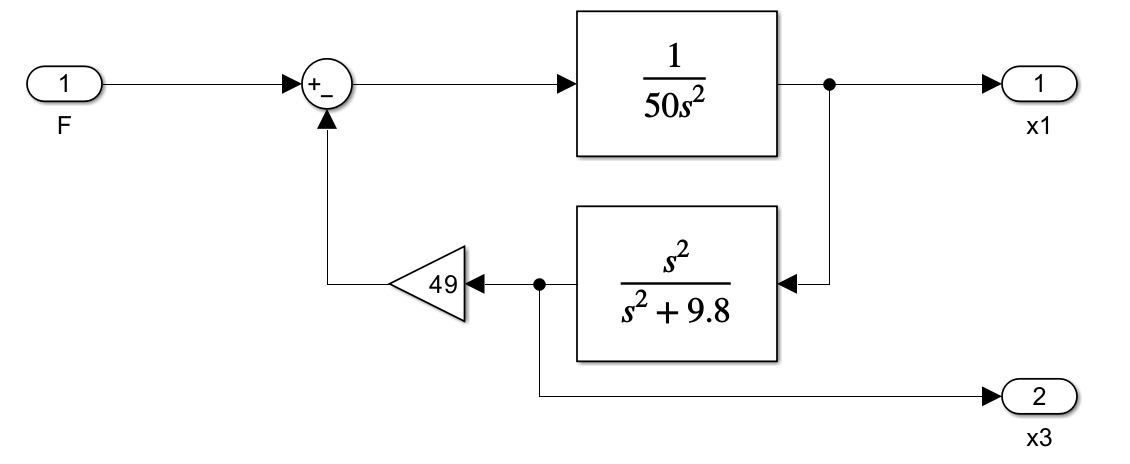


进一步代入便可得到

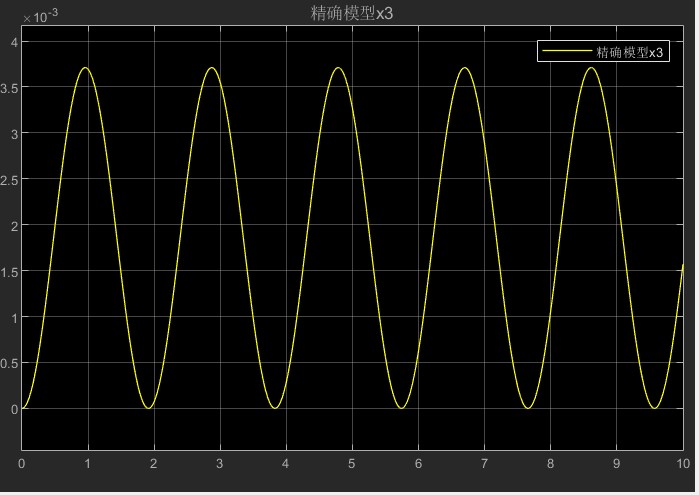
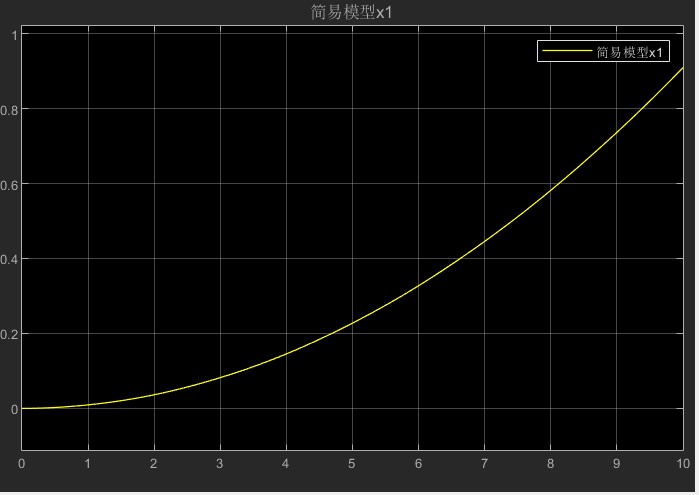


由此便可基于SIMULINK构建简易模型：

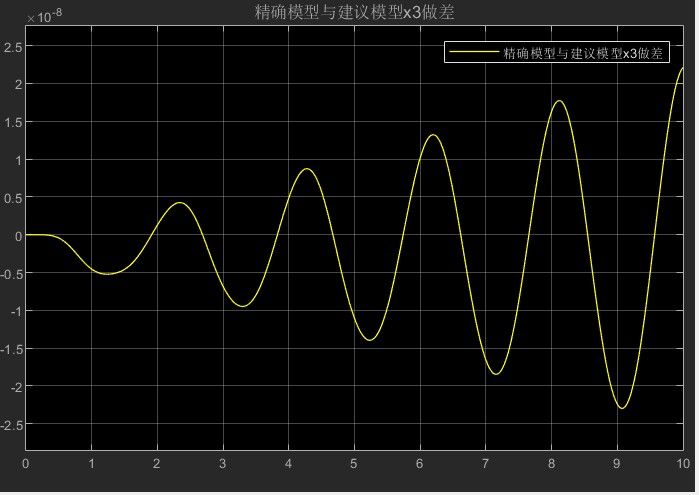
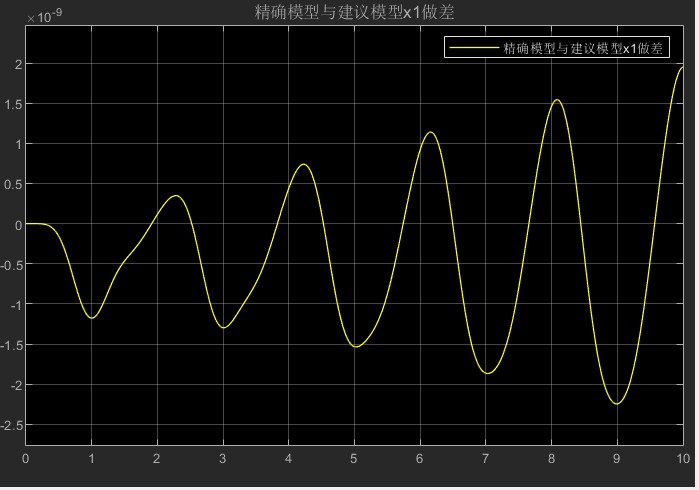




最终运行结果如下图所示：



### 复杂模型与简易模型对比



误差小于，说明简易模型能较好地与精确模型重合。

# 附录

|  |
| --- |
| 基础题一 |
| clear; clc; close all;  %% ode45  % 定义时间范围  tspan = [0 20];  % 初始条件 [x(0), x'(0)]  initial\_conditions = [1; 1];  % 求解微分方程  [t, state] = ode45(@system\_equation, tspan, initial\_conditions);  % 提取状态变量  x = state(:, 1); % 位置  x\_dot = state(:, 2); % 速度  % 计算加速度 (从微分方程推导)  x\_ddot = sin(pi\*t/2) - abs(x.^2 - 1).\*(x\_dot.^3) - x;  %% 绘制结果  % 位置曲线  figure;  plot(t, x, 'b-', 'LineWidth', 2);  grid on;  title('系统状态响应', 'FontSize', 14, 'FontWeight', 'bold');  ylabel('位置 x(t)', 'FontSize', 12);  legend('位置', 'Location', 'best');  xlim([0 20]);  % 速度曲线  figure;  plot(t, x\_dot, 'r-', 'LineWidth', 2);  grid on;  title('系统状态响应', 'FontSize', 14, 'FontWeight', 'bold');  ylabel('速度 dx/dt', 'FontSize', 12);  legend('速度', 'Location', 'best');  xlim([0 20]);  % 加速度曲线  figure;  plot(t, x\_ddot, 'g-', 'LineWidth', 2);  grid on;  title('系统状态响应', 'FontSize', 14, 'FontWeight', 'bold');  ylabel('加速度 d²x/dt²', 'FontSize', 12);  xlabel('时间 t (秒)', 'FontSize', 12);  legend('加速度', 'Location', 'best');  xlim([0 20]);  %% 绘制对比图  figure('Position', [100, 100, 1000, 600]);  plot(t, x, 'b-', 'LineWidth', 2);  hold on;  plot(t, x\_dot, 'r-', 'LineWidth', 2);  plot(t, x\_ddot, 'g-', 'LineWidth', 2);  grid on;  title('系统状态响应对比', 'FontSize', 16, 'FontWeight', 'bold');  xlabel('时间 t (秒)', 'FontSize', 12);  ylabel('状态变量', 'FontSize', 12);  legend('位置 x(t)', '速度 dx/dt', '加速度 d²x/dt²', 'Location', 'best');  xlim([0 20]);  %% 输出统计信息  fprintf('系统仿真结果统计:\n');  fprintf('仿真时间: 0 到 %.1f 秒\n', t(end));  fprintf('位置范围: [%.3f, %.3f]\n', min(x), max(x));  fprintf('速度范围: [%.3f, %.3f]\n', min(x\_dot), max(x\_dot));  fprintf('加速度范围: [%.3f, %.3f]\n', min(x\_dot), max(x\_dot));  %% 定义微分方程函数  function dstate = system\_equation(t, state)  x = state(1);  x\_dot = state(2);  F = sin(pi\*t/2);    nonlinear\_damping = abs(x^2 - 1) \* (x\_dot^3);    x\_ddot = F - nonlinear\_damping - x;    dstate = [x\_dot; x\_ddot];  end |

|  |
| --- |
| 基础题二 |
| % 力-质量系统仿真  % 运动方程: f - b\*dx/dt = M\*d²x/dt²  % 参数: f=1N, M=1kg, b=0.4N/(m/s)  % 拉力作用时间: 2秒  clear; clc; close all;  %% 系统参数  M = 1; % 质量 [kg]  b = 0.4; % 摩擦系数 [N/(m/s)]  f = 1; % 拉力 [N]  t\_pull = 2; % 拉力作用时间 [s]  sim\_time = 10; % 总仿真时间 [s]  %% ode45  % 定义时间范围  tspan = [0 sim\_time];  % 初始条件 [x(0), dx/dt(0)]  initial\_conditions = [0; 0]; % 假设初始位置和速度都为0  % 求解微分方程  [t, state] = ode45(@(t, y) mass\_system\_ode(t, y, M, b, f, t\_pull), tspan, initial\_conditions);  % 提取状态变量  x = state(:, 1); % 位置 [m]  v = state(:, 2); % 速度 [m/s]  % 计算加速度 [m/s²]  a = zeros(size(t));  for i = 1:length(t)  if t(i) <= t\_pull  a(i) = (f - b\*v(i)) / M;  else  a(i) = (-b\*v(i)) / M;  end  end  %% 绘制结果  % 位置曲线  figure;  plot(t, x, 'b-', 'LineWidth', 2);  grid on;  title('力-质量系统响应', 'FontSize', 14, 'FontWeight', 'bold');  ylabel('位置 x(t) [m]', 'FontSize', 12);  xlabel('时间 t [s]', 'FontSize', 12);  xlim([0 sim\_time]);  % 标记拉力作用结束时间  line([t\_pull t\_pull], ylim, 'Color', 'red', 'LineStyle', '--', 'LineWidth', 1);  text(t\_pull+0.1, max(x)\*0.8, '拉力停止', 'Color', 'red', 'FontSize', 10);  % 速度曲线  figure;  plot(t, v, 'r-', 'LineWidth', 2);  grid on;  ylabel('速度 v(t) [m/s]', 'FontSize', 12);  xlabel('时间 t [s]', 'FontSize', 12);  % 标记拉力作用结束时间  line([t\_pull t\_pull], ylim, 'Color', 'red', 'LineStyle', '--', 'LineWidth', 1);  text(t\_pull+0.1, max(v)\*0.8, '拉力停止', 'Color', 'red', 'FontSize', 10);  % 加速度曲线  figure;  plot(t, a, 'g-', 'LineWidth', 2);  grid on;  ylabel('加速度 a(t) [m/s²]', 'FontSize', 12);  xlabel('时间 t [s]', 'FontSize', 12);  xlim([0 sim\_time]);  % 标记拉力作用结束时间  line([t\_pull t\_pull], ylim, 'Color', 'red', 'LineStyle', '--', 'LineWidth', 1);  text(t\_pull+0.1, max(a)\*0.8, '拉力停止', 'Color', 'red', 'FontSize', 10);  %% 绘制在一张图中的对比图  figure('Position', [100, 100, 1000, 600]);  plot(t, x, 'b-', 'LineWidth', 2);  hold on;  plot(t, v, 'r-', 'LineWidth', 2);  plot(t, a, 'g-', 'LineWidth', 2);  grid on;  title('力-质量系统状态响应', 'FontSize', 16, 'FontWeight', 'bold');  xlabel('时间 t [s]', 'FontSize', 12);  % 标记拉力作用结束时间  line([t\_pull t\_pull], ylim, 'Color', 'black', 'LineStyle', '--', 'LineWidth', 1);  text(t\_pull+0.1, max(v)\*0.7, '拉力停止', 'Color', 'black', 'FontSize', 10);  legend('位置 x(t)', '速度 v(t)', '加速度 a(t)', 'Location', 'best');  %% 输出系统响应特性  fprintf('力-质量系统仿真结果:\n');  fprintf('========================\n');  fprintf('最大位置: %.4f m\n', max(x));  fprintf('最大速度: %.4f m/s\n', max(v));  fprintf('最大加速度: %.4f m/s²\n', max(a));  fprintf('最终位置: %.4f m\n', x(end));  fprintf('最终速度: %.4f m/s\n', v(end));  fprintf('系统达到最大速度的时间: %.2f s\n', t(v == max(v)));  %% 定义微分方程函数  function dstate = mass\_system\_ode(t, state, M, b, f, t\_pull)  % state(1) = x (位置)  % state(2) = dx/dt (速度)    x = state(1);  v = state(2);    % 分段外力函数：前2秒有拉力，之后无拉力  if t <= t\_pull  F = f;  else  F = 0;  end    % 计算加速度  a = (F - b\*v) / M;    % 返回状态导数 [dx/dt; dv/dt]  dstate = [v; a];  end |

|  |
| --- |
| 基础题四 |
| clear; clc; close all;  %% 系统参数  mu = 1/82.45; % μ值  mu\_star = 1 - mu; % μ\*值  %% 初始条件  x0 = 1.2; % 初始x位置  dx0 = 0; % 初始x速度  y0 = 0; % 初始y位置  dy0 = -1.04935751; % 初始y速度  initial\_conditions = [x0; dx0; y0; dy0];  %% 仿真时间设置  t\_span = [0 10]; % 仿真时间范围 [0, 10] 单位时间  %% 求解微分方程组  [t, state] = ode45(@(t, y) spacecraft\_equations(t, y, mu, mu\_star), t\_span, initial\_conditions);  %% 提取状态变量  x = state(:, 1); % x位置  dx = state(:, 2); % x速度  y = state(:, 3); % y位置  dy = state(:, 4); % y速度  %% 计算加速度 - 修复后的方法  ddx = zeros(size(t));  ddy = zeros(size(t));  for i = 1:length(t)  dstate = spacecraft\_equations(t(i), state(i,:)', mu, mu\_star);  ddx(i) = dstate(2);  ddy(i) = dstate(4);  end  %% 绘制轨迹图  % 主图：飞行器轨迹  figure;  plot(x, y, 'b-', 'LineWidth', 1.5);  hold on;  % 标记起点和终点  plot(x(1), y(1), 'go', 'MarkerSize', 8, 'MarkerFaceColor', 'g');  plot(x(end), y(end), 'ro', 'MarkerSize', 8, 'MarkerFaceColor', 'r');  grid on;  axis equal;  title('空间飞行器运动轨迹', 'FontSize', 14, 'FontWeight', 'bold');  xlabel('x 坐标', 'FontSize', 12);  ylabel('y 坐标', 'FontSize', 12);  legend('飞行轨迹', '起点', '终点', 'Location', 'best');  % 添加参数信息  text(0.02, 0.98, sprintf('μ = %.6f\nμ\* = %.6f', mu, mu\_star), ...  'Units', 'normalized', 'VerticalAlignment', 'top', ...  'BackgroundColor', 'white', 'EdgeColor', 'black', 'FontSize', 10);  % x位置随时间变化  figure;  plot(t, x, 'r-', 'LineWidth', 1.5);  grid on;  title('x位置随时间变化', 'FontSize', 12);  xlabel('时间', 'FontSize', 10);  ylabel('x坐标', 'FontSize', 10);  % y位置随时间变化  figure;  plot(t, y, 'g-', 'LineWidth', 1.5);  grid on;  title('y位置随时间变化', 'FontSize', 12);  xlabel('时间', 'FontSize', 10);  ylabel('y坐标', 'FontSize', 10);  %% 绘制速度变化图  figure;  plot(t, dx, 'b-', 'LineWidth', 1.5);  grid on;  title('x方向速度变化', 'FontSize', 12);  xlabel('时间', 'FontSize', 10);  ylabel('dx/dt', 'FontSize', 10);  figure;  plot(t, dy, 'm-', 'LineWidth', 1.5);  grid on;  title('y方向速度变化', 'FontSize', 12);  xlabel('时间', 'FontSize', 10);  ylabel('dy/dt', 'FontSize', 10);  %% 绘制3D轨迹图  figure('Position', [100, 100, 800, 600]);  plot3(x, y, t, 'b-', 'LineWidth', 1.5);  hold on;  plot3(x(1), y(1), t(1), 'go', 'MarkerSize', 8, 'MarkerFaceColor', 'g');  plot3(x(end), y(end), t(end), 'ro', 'MarkerSize', 8, 'MarkerFaceColor', 'r');  grid on;  title('空间飞行器轨迹 (x-y-t)', 'FontSize', 14, 'FontWeight', 'bold');  xlabel('x坐标', 'FontSize', 12);  ylabel('y坐标', 'FontSize', 12);  zlabel('时间', 'FontSize', 12);  legend('飞行轨迹', '起点', '终点', 'Location', 'best');  %% 输出系统信息  fprintf('空间飞行器运动仿真结果:\n');  fprintf('========================\n');  fprintf('参数: μ = %.6f, μ\* = %.6f\n', mu, mu\_star);  fprintf('初始条件: x(0)=%.2f, dx/dt(0)=%.2f, y(0)=%.2f, dy/dt(0)=%.8f\n', ...  x0, dx0, y0, dy0);  fprintf('仿真时间: %.1f 到 %.1f\n', t\_span(1), t\_span(2));  fprintf('轨迹范围: x ∈ [%.3f, %.3f], y ∈ [%.3f, %.3f]\n', ...  min(x), max(x), min(y), max(y));  fprintf('最终位置: (%.4f, %.4f)\n', x(end), y(end));  %% 定义微分方程函数  function dstate = spacecraft\_equations(t, state, mu, mu\_star)  % state = [x; dx/dt; y; dy/dt]  x = state(1);  dx = state(2);  y = state(3);  dy = state(4);    % 计算 r₁ 和 r₂  r1 = sqrt((x + mu)^2 + y^2);  r2 = sqrt((x - mu\_star)^2 + y^2);    % 计算加速度项  ddx = 2\*dy + x - (mu\_star\*(x + mu))/(r1^3) - (mu\*(x - mu\_star))/(r2^3);  ddy = -2\*dx + y - (mu\_star\*y)/(r1^3) - (mu\*y)/(r2^3);    % 返回状态导数  dstate = [dx; ddx; dy; ddy];  end |