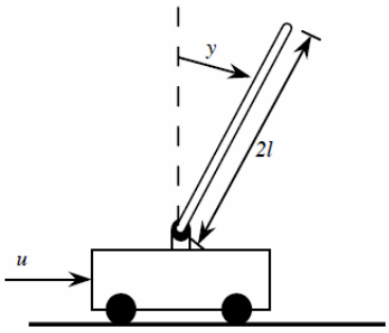
智能控制 作业 2 专家控制 代码实现报告

题目

专家控制作业

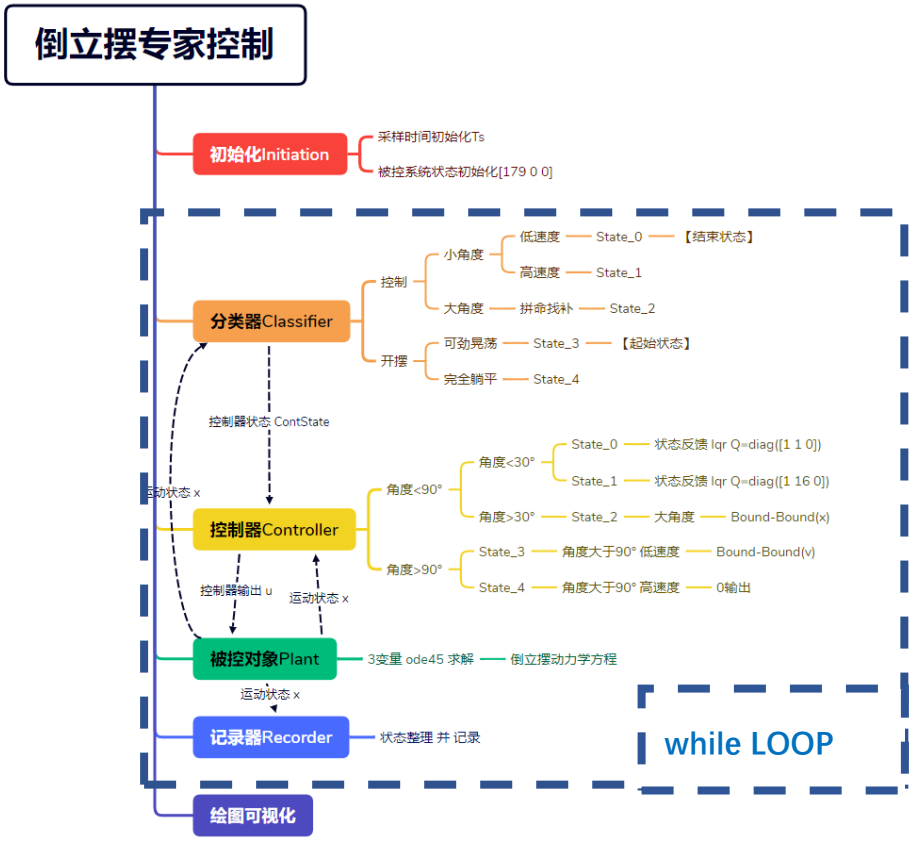


若车载倒立摆系统的模型为：

$$\ddot{y} = \frac{9.8\sin(y) + \cos(y)\left[\frac{-\bar{u}-0.25\dot{y}^2\sin(y)}{1.5}\right]}{0.5\left[\frac{4}{3} - \frac{1}{3}\cos^2(y)\right]}$$
$$\dot{\bar{u}} = -100\bar{u} + 100u.$$

其中 u 为施加在车上的推拉力，y 为倒立摆与竖直线的角度。若系统初始时刻，角度为 0.1 弧度，角速度为 0 弧度/秒，期望的控制目标是使倒立摆的角度为零，请设计专家 PID 控制器并进行仿真。

实现思路和整体架构：

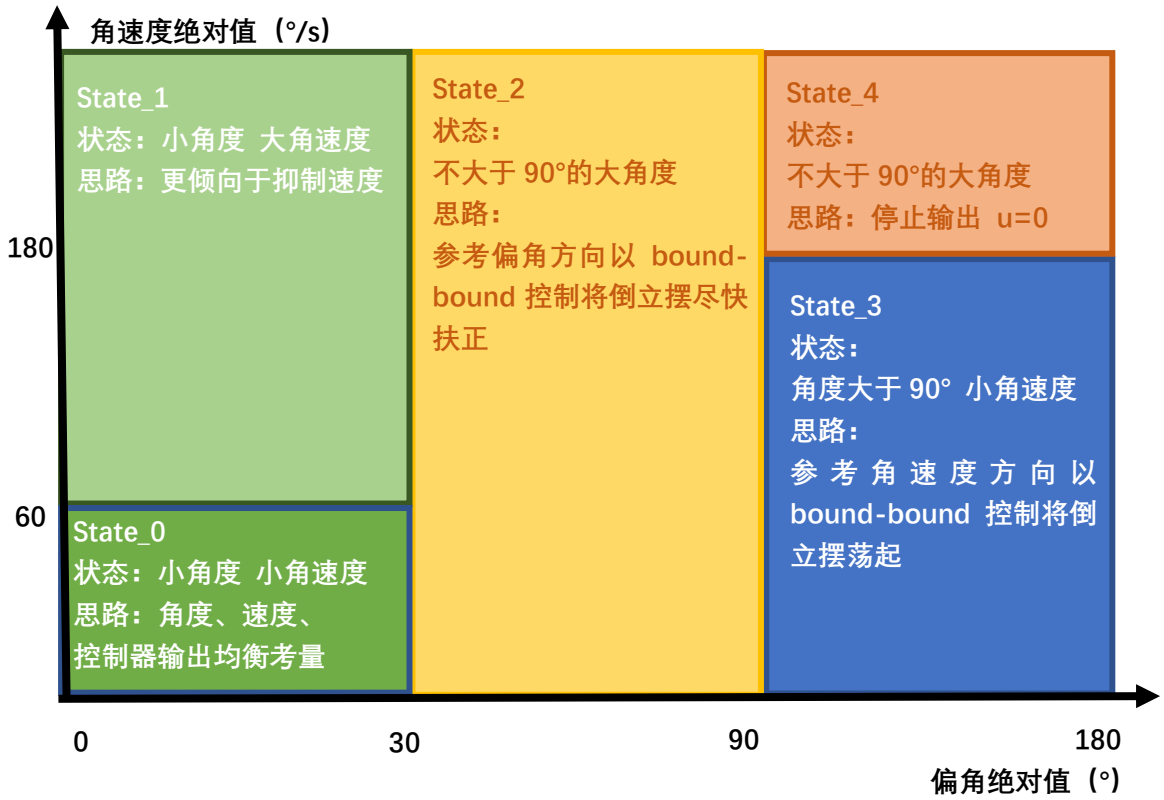


如上图所示，在初始化完成后，程序进入 while 循环，依次调用分类器、控制器和动力学模型相关函数，依次进行控制器状态分类、状态对应控制器控制和被控对象响应。

其中分类器由以倒立摆偏角和角速度为条件的 5 条判断分支组成，通过返回值输出至控制器；控制器根据分类器提供的状态分类按不同逻辑输出，控制器输出值认为可以瞬间改变且在两次采样间不变，以函数返回值输出至被控对象动力学模型；被控对象套用题目中所给的倒立摆动力模型，将 y、dy/dt、u_分别作为状态变量 x1 x2 x3，以 ode45 求解其在每两次采样间的变化过程，通过函数返回值输出；X 的值在每次循环中被组织记录，以便此后分别绘制时域图和状态轨迹，并作为分类依据输入下一轮循环的分类器。

下面主要介绍分类器和对应控制器的设计思路。

分类器与控制器各状态说明

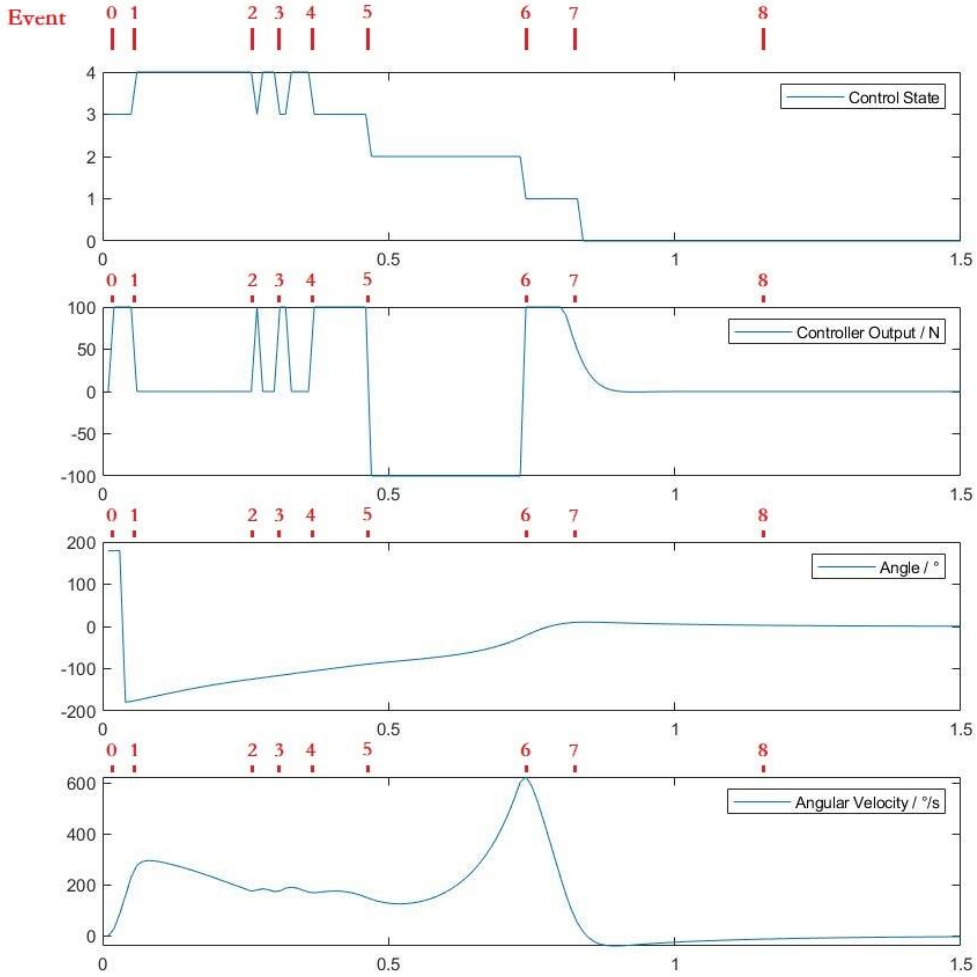


上图所示即为在x(1)-x(2)平面(x(1)对应摆杆偏角、x(2)对应摆杆角速度)其中，状态state_0和状态 state_1 的控制器设计是对系统进行小角近似后，以 LQR 方法构造的状态反馈。在 state_0 中，将 Q 设为对角线元素依次为 1、1、0（分别对应摆杆偏角、摆杆偏角角速度，间接控制量 u_）的对角阵，R 设置为 1；在 state_1 中，将 Q 设为对角元素依次为 1、16、0 的对角阵，R 设置为 1。与 state_0 相比，state_1 中通过将 Q 中第 2 个对角元扩大，更倾向于配置抑制角速度的状态反馈。

在 state_2 和 state_3 中，控制器均采用 bound-bound 控制。在 state_2 中认为摆杆已经倾斜到比较大的角度，需要 u 以控制器最大输出（在本例中，将最大输出力设为了 100N）来防止摆杆倾倒，其中控制器输出量的正负取决于摆杆偏角的正负；在 state_3 中认为摆杆基本处于下垂状态且摆动速度较慢，需要施加激励使其开始较快摆动后竖起。

在仅有 state0-3 的控制器的情况下，进行了仿真测试，发现此时摆杆在一些特殊情况下会出现以过快速度通过竖直状态，而 state_1 的控制器无法有效降低摆杆速度的情况。于是在摆杆偏角大于 90°且角速度大于 180°/s 时增加了一个状态 state_4，在 state_4 中控制器输出恒为 0。在这一状态下，如果摆的偏角减小，即摆的重心上升，摆的速度会有所降低，此时再跳转至 state_3 进行 bound-bound 输出。通过这一状态的设置，保证了摆杆通过正负 90°偏角的角速度上限，避免了上述问题的发生

程序仿真结果



上图是使用设计的专家系统控制倒立摆从自然下垂位置(实际设置为 179°)零角速度初始到最终稳定直立的控制过程。

以下是对上图中标注的各个状态变化的时间节点做简单分析说明:

Event 0

控制过程开始，初始角度为 179°，初始角速度为 0，可以认为在基本自然下垂状态加入一个微小扰动。摆杆自这一角度释放后摆过最低点跳变为-180°并产生了较小的正向的角速度，此时摆杆低于水平面且摆动速度较慢，分类器作 state_3，控制器与角速度同号进行 bound-bound 输出扩大摆角；

Event 1

摆杆角速度首次达到预设阈值 180°/s, 此时摆杆角度未高于水平, 分类器跳转至 state_4, 控制器输出归零，摆杆以状态切换时的角度和角速度为初始条件做自由上摆，角速度略大于设定值；

Event 2 Event 3

随着摆杆摆高偏角缩小，系统动能转化为重力势能，摆杆角速度下降。当摆杆角速度低于预设阈值 180°/s 时，分类器再次跳转 state_3，控制器 bound-bound 输出，加速摆杆运动；

Event 4

摆高继续摆高，偏角继续减少，摆杆相对于转轴的重力矩不断增加而可以产生的最大惯性矩不断减少。直至 Event 4 之后，即使分类器作 state_3 而控制器以最大 bound-bound 持续输出也不能以惯性矩克服重力矩，摆杆角速度逐渐下降；

Event 5

摆角在惯性作用下摆过 90°，高于水平，分类器跳转 state_2，开始以偏角正负为参考的 bound-bound 控制，迅速减小偏角；

Event 6

偏角在 bound-bound 控制下迅速减小到 30°，此时摆杆角速度大于预设阈值 60°/s，分类器跳转 state_1，开始优先抑制角速度（仍然出现了大段控制器输出饱和，改一改效果可以更理想）；

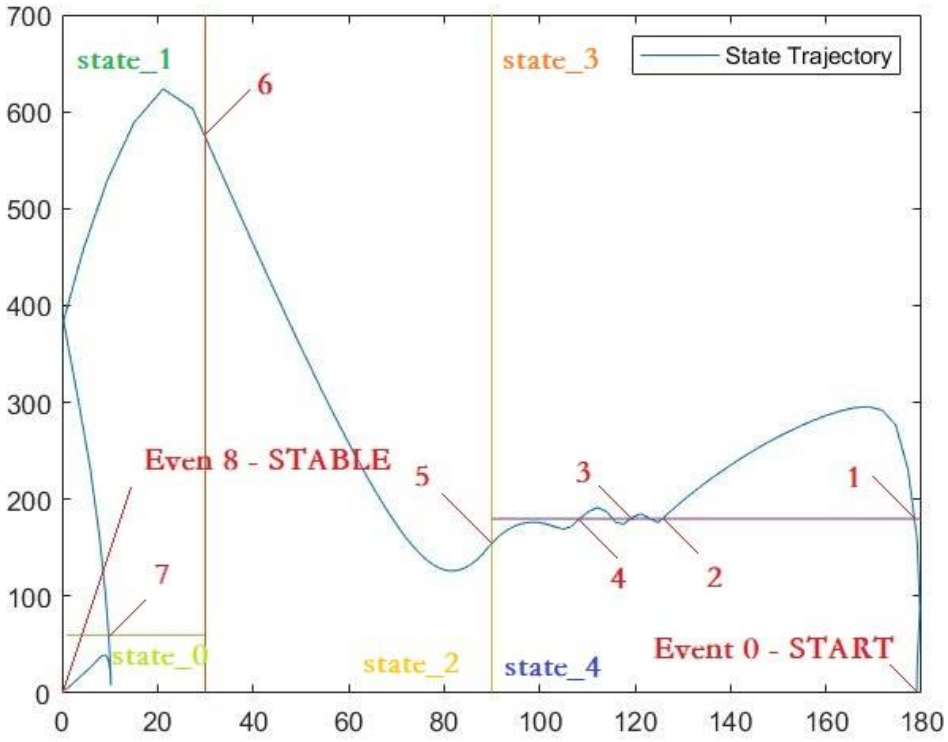
Event 7

摆杆角速度在状态反馈控制器作用下逐渐降低至小于预设阈值 60°/s，分类器跳转至 state_0，开始优先抑制角度。

Event 8

可以认为此时系统趋于稳定，整个过程耗时约 1.2 s。

从状态空间讨论



上图为仿真结果在状态空间下的表示，其横坐标为倒立摆偏角绝对值，单位为°，纵坐标为角速度绝对值，单位为°/s。在时域中讨论的各个状态分类器跳变节点在仿真结果的状态空间轨迹上也可以找到对应位置，这说明时域上反映出来的状态分类机跳变是可靠的。上图中，各个跳变节点的意义就不再一一重复讨论了。

从中可以看出在时域上 Event 6-Event 7 之间出现的控制器输出饱和的主要原因在于 Event 5-Event 6 之间摆杆角速度快速增加。针对这一问题可以提出两个解决方案：1.将 state_0 和 state_1 同 state_2 之间的角度阈值适度加大，取 45°或 60°等角度，可能可以避免 bound-bound 控制导致的角速度过大 2.将 state_2 中角速度大于 180°/s 上方的区域划入 state_3，即在角速度过大是亦将输出置 0。

总结

在本次作业中，我搭建了题目中倒立摆的仿真框架，并设计了以偏角角度、角速度为分类依据，对于 5 个不同状态分别采取了两种不同的状态反馈控制、根据偏角角度和角速度符号的 bound-bound 控制和控制器 0 输出使动力学系统自由发展的专家控制方案，并分别从时域和状态空降两个角度讨论了仿真结果状态分类跳转的正确性。

这次作业主要有两点遗憾：第一是仿真中 Even6-Event7 过程中的控制器输出饱和问题虽然提出想法和思路但没有解决，不够优雅成功；第二是没看清原题目中对于 0.1rad 为初始条件和 PID 控制（可能更多讨论在通过不同状态的切换实现性能优化这方面）的限制，而采用了状态反馈设计了从自由下垂到竖起的整个控制过程(其实在小角度高速时优先抑制速度或许也是一种提高性能的讨论？只是在具体每个状态下的控制器用的不是 PID 而是状态反馈)。

但无论如何，在这次的过程中我还是通过实践理解了专家控制通过反馈量的阈值划分不同状态，采取不同控制方案以提高性能的概念，希望下次有机会做得更好吧。

（后附代码）

expertSystem.m

```
%   Intelligent Control ZJU 2022-FALL HW2
%   Expert System
%   created by StvLi LiPZ 20221202
clear;
clc;
%   Part0 Initiation
%       run in 1 time
%   simulation time
Tt  = 1.5;

%   sample time
Ts  = 0.01;

%   initial state
%   x1 - angle y
%   x2 - angular velocity dy
%   x3 - force u_
x   = [179*pi/180 0 0]';

%   allocate recording space
stepToRun   = Tt/Ts;
precStep    = 0;
ur  = zeros( 1 , stepToRun );
y   = zeros( 2 , stepToRun );
sr  = zeros( 1 , stepToRun );

while ( precStep*Ts<Tt )   %   run in loop
    %   Classifier
    ContState = myExpeContClassifier( x );
    sr(precStep+1) = ContState;

    %   Controller
    u   = myExpeContController( x , ContState );

    %   Plant
    x = myPlantDyna( x , u , Ts );

    %   Recorder
    precStep = precStep+1;

    y(1,precStep)   = x(1);
```

```
    y(2,precStep)   = x(2);
    ur(1,precStep)  = u;
    t(1,precStep)   = precStep * Ts;
end
disp('Simulation DONE');

%   plot
%       time domain
subplot(4,1,1);
plot(t,sr);
legend('Control State');

subplot(4,1,2);
plot(t,ur);
legend('Controller Output / N');

subplot(4,1,3);
plot(t,y(1,:)*180/pi);
legend('Angle / ¡ã');

subplot(4,1,4);
plot(t,y(2,:)*180/pi);
legend('Angular Velocity / ¡ã/s');

% %       state space
% plot(abs(y(1,:)*180/pi),abs(y(2,:)*180/pi))
% hold on
% plot(30*ones(700,1),1:700)
% hold on
% plot(90*ones(700,1),1:700)
% hold on
% plot(90:180,180*ones(91,1))
% hold on
% plot(1:30,60*ones(30,1))
% legend('State Trajectory');
```

myExpeContClassifier.m

```
%   My Classifier used in Expert Control
%       created by StvLi 20221202

%       input: 3 state coorinates
%       x1  angle
%       x2  angular velocity
%       x3  force

%       output:
%       3 control states
%           0   little angle slow velocity
%           1   little angle fast velocity
%           2   large angle
%       2 start states
%           3   slow velocity
%           4   fast velocity

function ContState = myExpeContClassifier( x )
%   Threshold Value Steup
angleThre_0      = 90/180*pi  ;    %   Start or Control
angleThre_1      = 30/180*pi  ;    %   Control:    small or
large angle
anguVeloThre_0    = 180/180*pi ;    %   Start:      slow or
fast velocity
anguVeloThre_1    = 60/180*pi  ;    %   Control:    slow or
fast velocity

    if(abs(x(1)) < angleThre_0)
        if ( abs(x(1)) < angleThre_1)
            if( abs(x(2)) < anguVeloThre_1 )
                ContState = 0 ;
            else
                ContState = 1 ;
            end
        else
            ContState = 2 ;
        end
    else
        if( abs(x(2)) < anguVeloThre_0 )
            ContState = 3 ;
        else
```

```
            ContState = 4 ;
        end
    end
%     disp('Classifier DONE State: ');
%     disp(ContState);
end
```

myExpeContController.m

```
%   My Controller used in Expert Control
%       created by StvLi 20221202

%       input: 3 state coorinates & Control State
%       x1  angle
%       x2  angular velocity
%       x3  force
%       ContState

%       output:
%       u   output value
function u = myExpeContController( x , ContState )
    boundary = 100;
    switch( ContState )
        %       0   little angle slow velocity
        case 0
            A = [
                0       1   0   ;
                19.6    0   0   ;
                0       0   -100;
                ];
            B = [ 0 -0.444 100]';
            Q = diag([1 1 0]);
            R = 1;
            [K,~,~] = lqr(A,B,Q,R);
            u = -K*x;
            if( u > boundary )
                u = boundary;
            end
            if(u < -boundary )
                u = -boundary;
            end
%             disp("Controller DONE State: 0");

        %       1   little angle fast velocity
        case 1
            A = [
                0       1   0   ;
                19.6    0   0   ;
                0       0   -100;
                ];
```

```
            B = [ 0 -0.444 100]';
            Q = diag([1 16 0]);
            R = 9;
            [K,~,~] = lqr(A,B,Q,R);
            u = -K*x;
            if( u > boundary )
                u = boundary;
            end
            if(u < -boundary )
                u = -boundary;
            end
%             disp("Controller DONE State: 1");

        %       2   large angle
        case 2
            u = sign(x(1))*boundary;
%             disp("Controller DONE State: 2");
        case 3
            u = sign(x(2))*boundary;
        case 4
            u = 0 ;
        otherwise
            disp("ERROR STATE!")
    end

end
```

myPlantDyna.m

```
%   My Plant Dynamics Model used in Expert Control
%       created by StvLi 20221202

%       input:
%       x1  angle
%       x2  angular velocity
%       x3  force
%       u   control value
%       Ts  sampling time

%       output:
%       x1  angle
%       x2  angular velocity
%       x3  force
function x = myPlantDyna( x , u , Ts )
    x_temp = zeros(1,3);

    [ ~ , x_temp ] = ode45(@invePendDyna,[0 Ts],[ x(1)
x(2) x(3) ]');
    function dx_temp = invePendDyna(~,x_temp)
        dx_temp = [ x_temp(2);
                    (9.8*sin(x_temp(1))+cos(x_temp(1))*...
                    (-x_temp(3)-
0.25*(x_temp(2)^2*sin(x_temp(1))))/1.5)/...
                    (0.5*(4/3-1/3*cos(x_temp(1))^2));
                   -100*x_temp(3) + 100*u;
                    ];
    end
    % Angle Regulation: [-pi,pi]
    while (x_temp(end,1) > pi)
        x_temp(end,1) = x_temp(end,1)-2*pi;
    end
    while (x_temp(end,1) < -pi)
        x_temp(end,1) = x_temp(end,1)+2*pi;
    end

    x(1) = x_temp(end,1);
    x(2) = x_temp(end,2);
    x(3) = x_temp(end,3);
%     disp('Plant DONE');
end
```