Corresponding ki, kd, kp

0, 0, 1,
0, 0, 0.01,
0, 0, 0.05,
0.005, 0.1, 0.05,
0.005, 0.5, 0.05,
0.5, 0.1, 0.05,

Corresponding performance

Overshoot for row 1,2,3,4,5,6(degrees):
35 0 0 19 28 36
Steady Error for row 1,2,3,4,5,6(degrees):
0 26 0 0 -1 0
Rise time for row 1,3,4,5,6(degrees):
257ms 534ms 273ms 259ms 544ms
Settling time for row 1,2,3,4,5,6:
1335ms 2064ms 273ms 582ms 606ms 6580ms

Codes

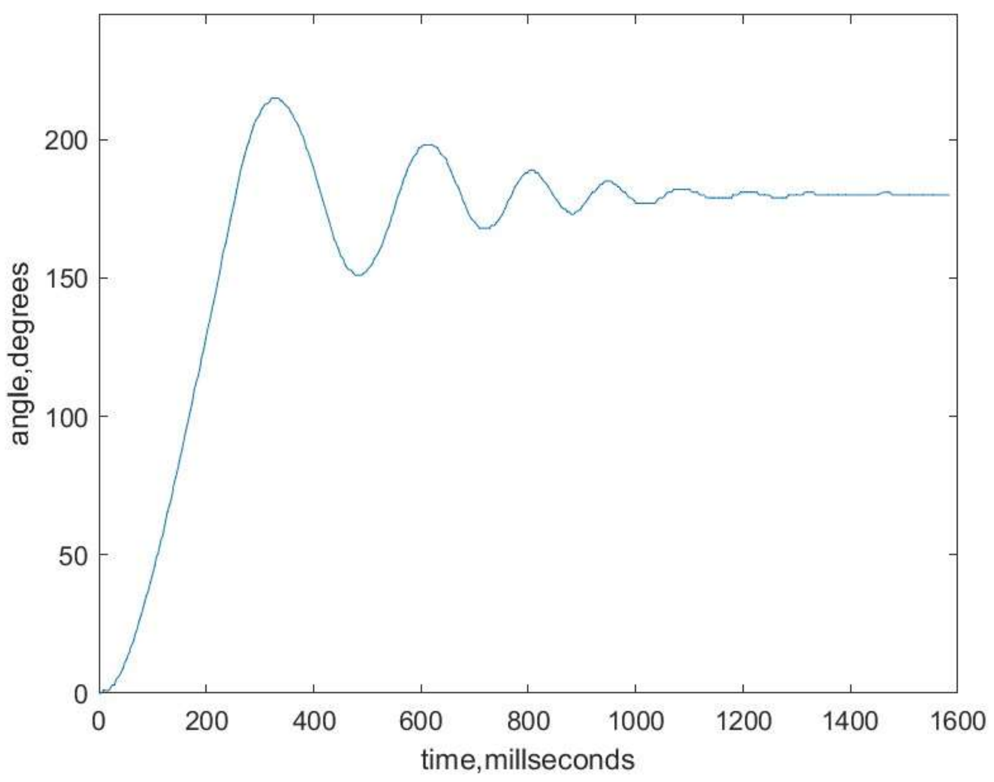
```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor,
GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile


# This program requires LEGO EV3 MicroPython v2.0 or higher.
# Click "Open user guide" on the EV3 extension tab for more
information.


# Create your objects here.
ev3 = EV3Brick()

m = Motor(port=Port.D)
sw = StopWatch()
```

```
data = DataLog(append=True,name="Data1")

def PID(past_time,present_time,target,real,ki,kp,kd,anti_over,range):
    error = target - real
    i = ki*error*(present_time-past_time)
    p = kp*error
    d = kd*error/(present_time-past_time)
    rvalue = p+i+d
    if (anti_over == False):
        return rvalue
    elif (abs(rvalue)<range):
        return rvalue
    else :
        return rvalue/abs(rvalue)*range

MAX_TIME = 10
time_present = 0
TARGET = 180

ev3.speaker.say("Beep,I'm going to run!")
data.log("target",TARGET)
data.log("time_mill","angle")

while True:
    time_last = time_present
    time_present = sw.time()
    data.log(time_present,m.angle())
    if (time_present>MAX_TIME*1000):
        break
    m.dc(PID(past_time=time_last,
             present_time=time_present,
             target = TARGET,
             real = m.angle(),
             ki=0.5,kd=0.1,kp=0.05,
             anti_over=True,
             range=7)
        /7*100)
```

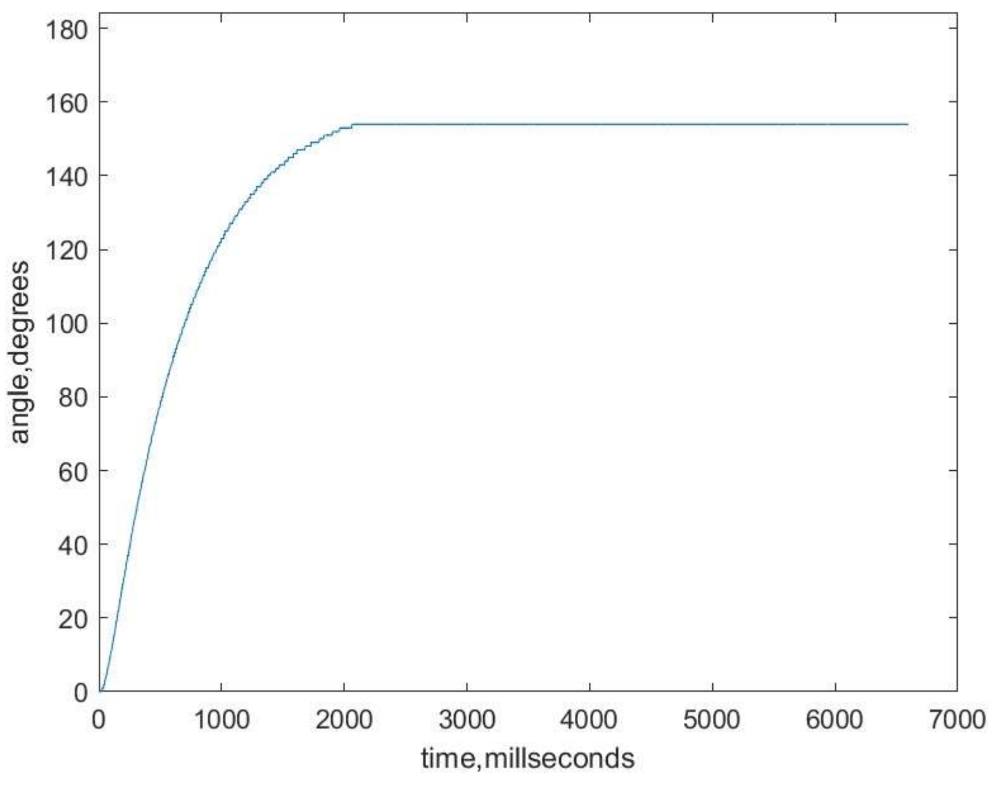
```
kdata = readmatrix("k_data");
k = 2.0;
Tm = 0.6;
```

```
for i = 1:6
    data = readmatrix("Data"+i);
    target = data(1,2);
    ki = kdata(i,1);
    kd = kdata(i,2);
    kp = kdata(i,3);
    if i == 2 || i >=3
        stedyerr(i) = data(1,2) - data(end,end);
    end
    figure('Name','angle-t_graph');
    grid on;
    grid minor;
    plot(data(3:end,1)-data(3,1),data(3:end,2));
    maxangle = max(data(3:end,2));
    if i == 1 || i >=3
        oversangle(i) = maxangle - data(1,2);
    end
    ylim([0,maxangle+30]);
    hold on;
    xlabel('time,millseconds');
    ylabel('angle,degrees');
    si = sim('call_model');
    plot(si.yout{1}.Values,'r');
    print('compare'+"pic_data"+string(i),'-djpeg');
end
close all;
```

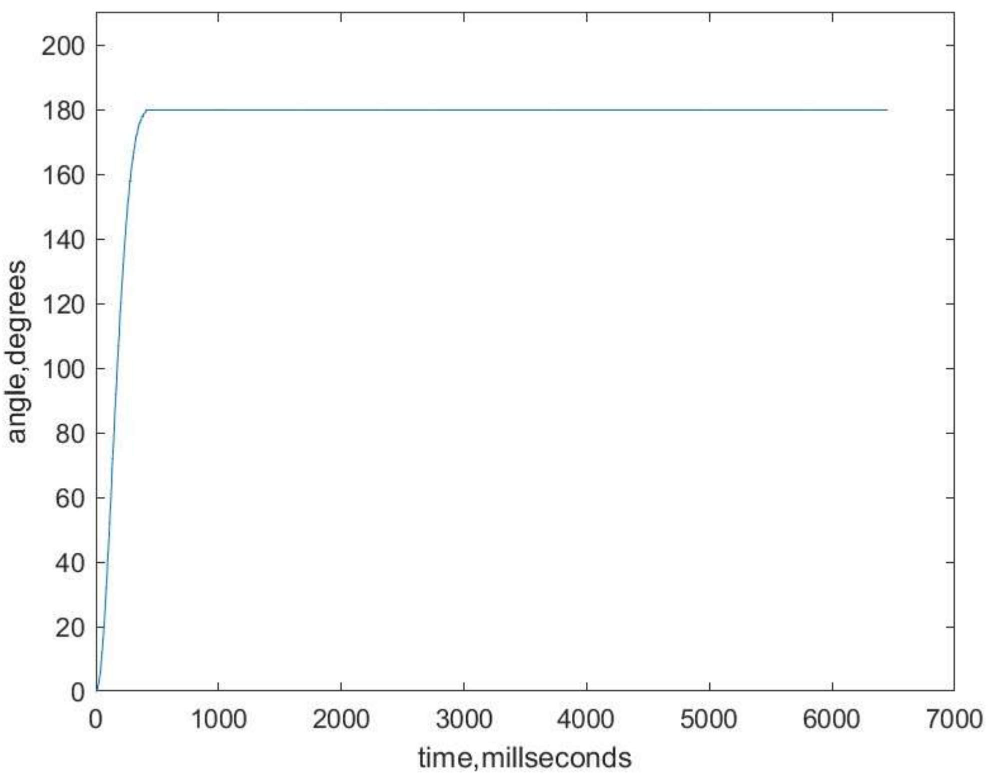
Real Data



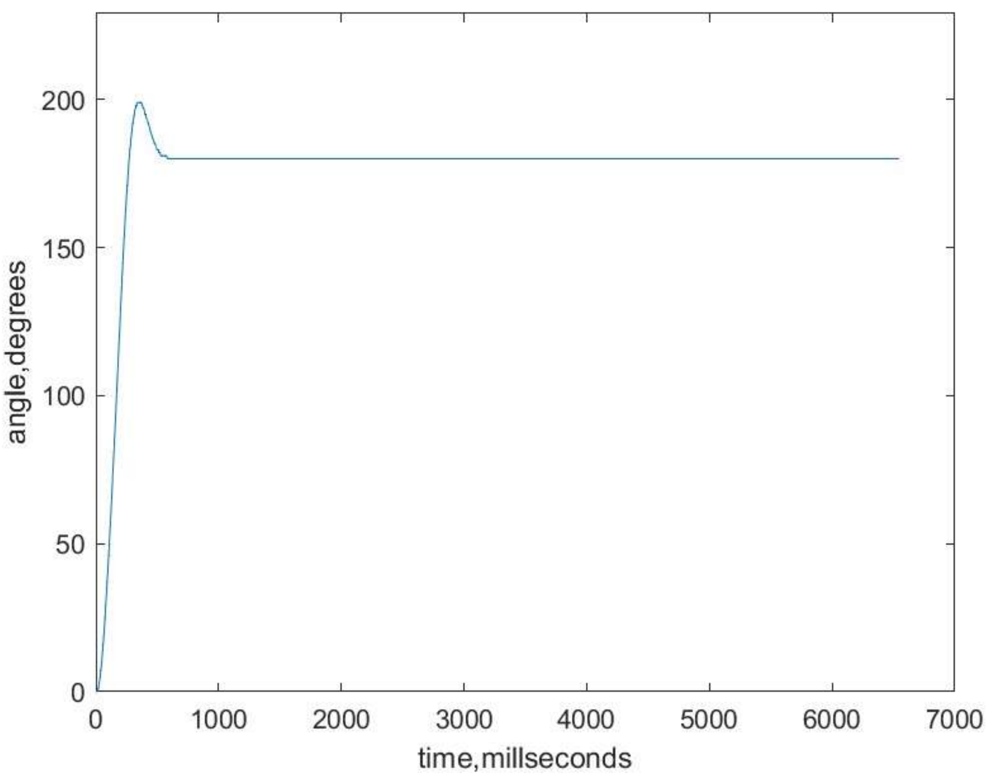
200
150
100
50
0
angle,degrees
0
200
400
600
800
1000
1200
1400
1600
time,millseconds



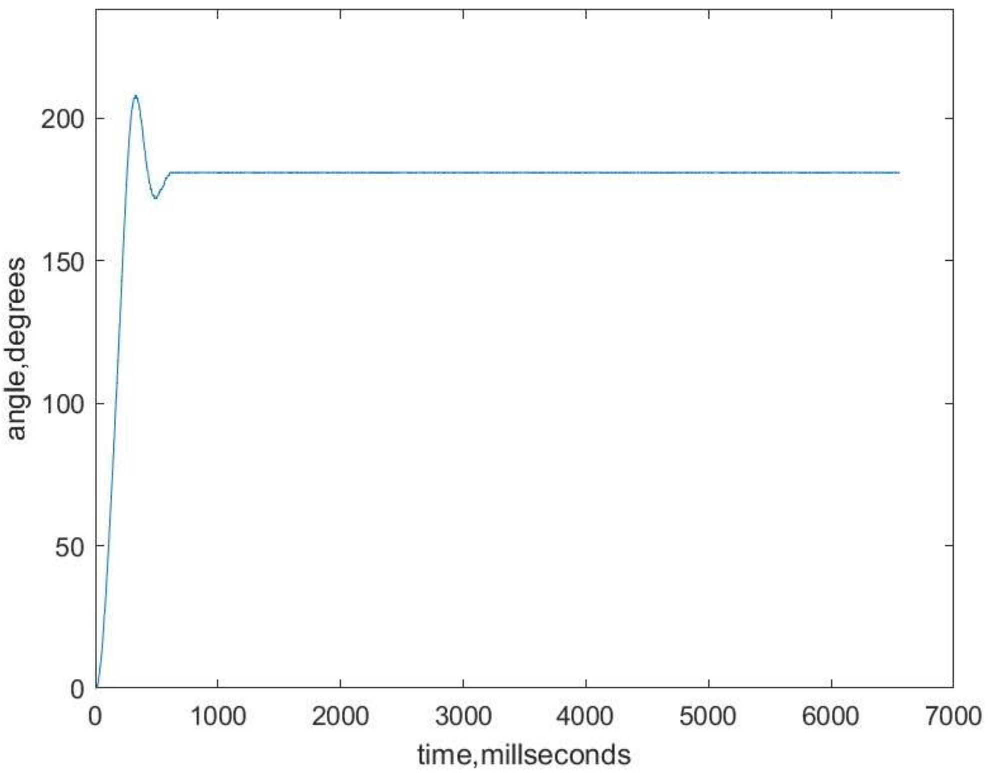
180
160
140
120
100
80
60
40
20
0
angle,degrees
0
1000
2000
3000
4000
5000
6000
7000
time,millseconds

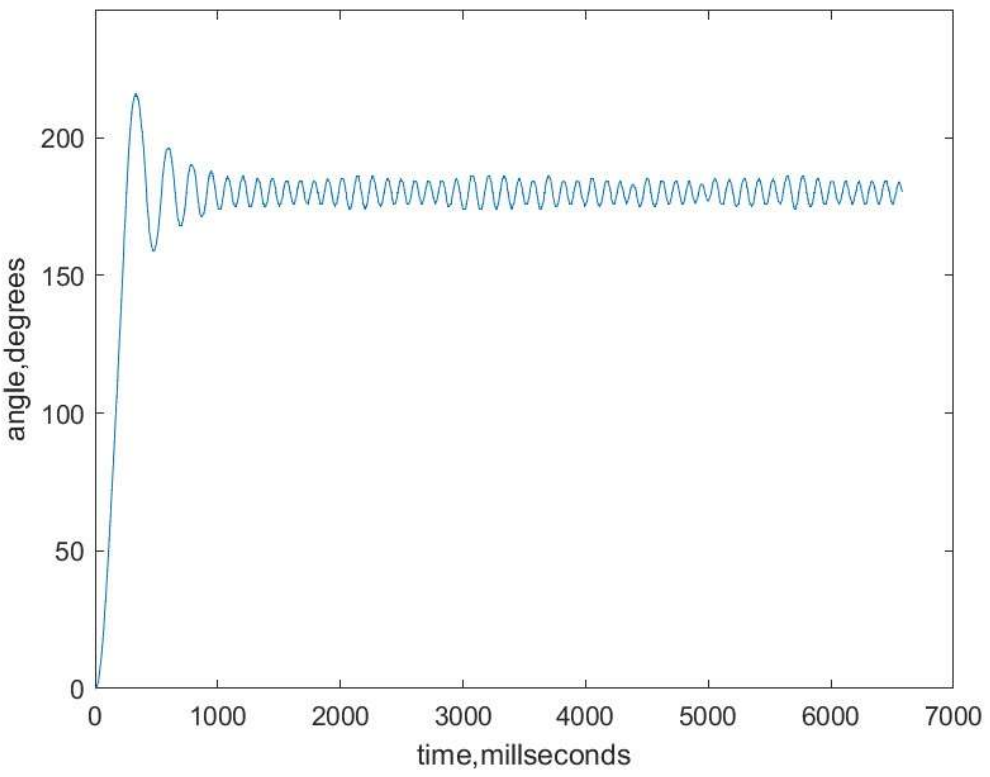


angle,degrees
time,millseconds

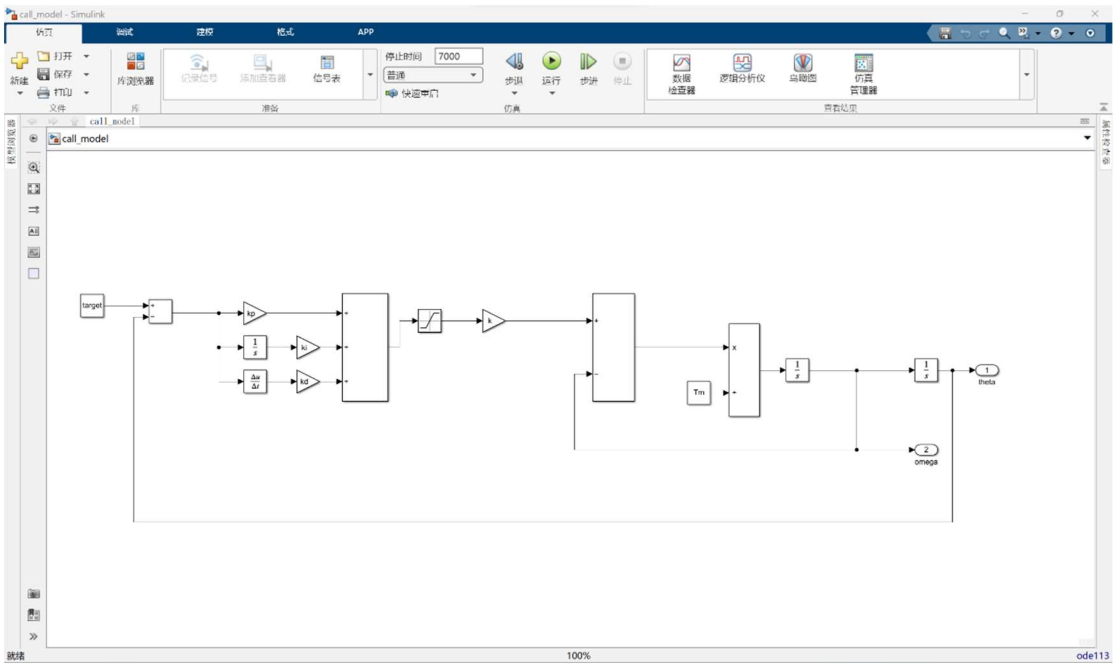


angle,degrees
time,millseconds


angle,degrees
time,millseconds
0
50
100
150
200
1000
2000
3000
4000
5000
6000
7000

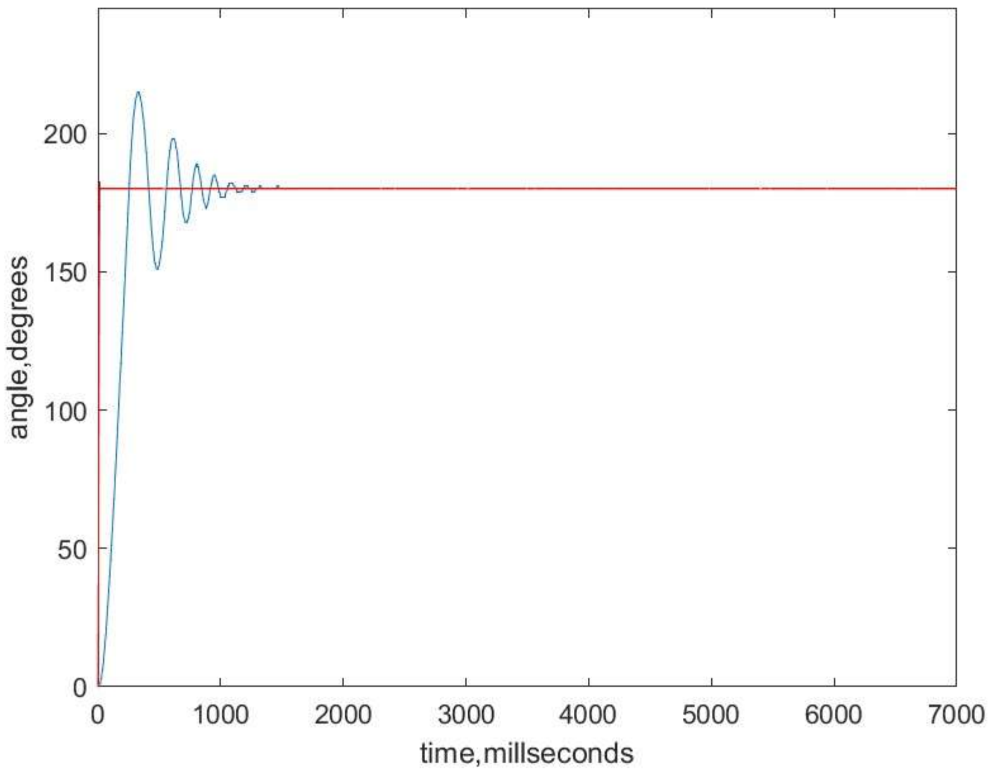

angle,degrees
time,millseconds
0
50
100
150
200
1000
2000
3000
4000
5000
6000
7000

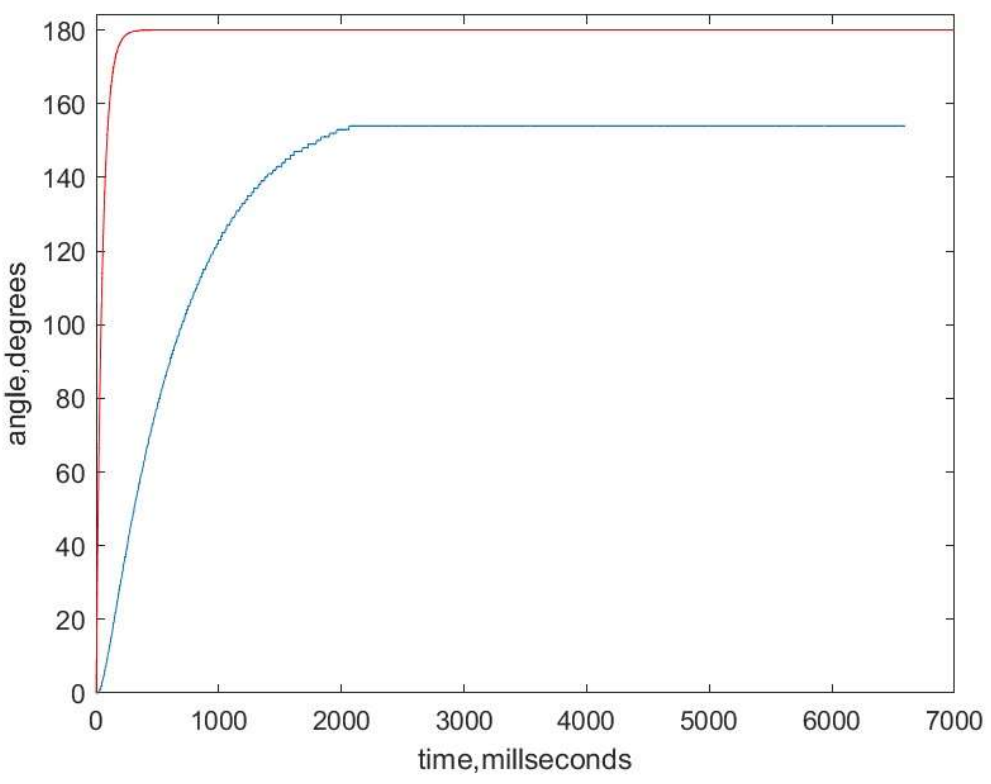
Model



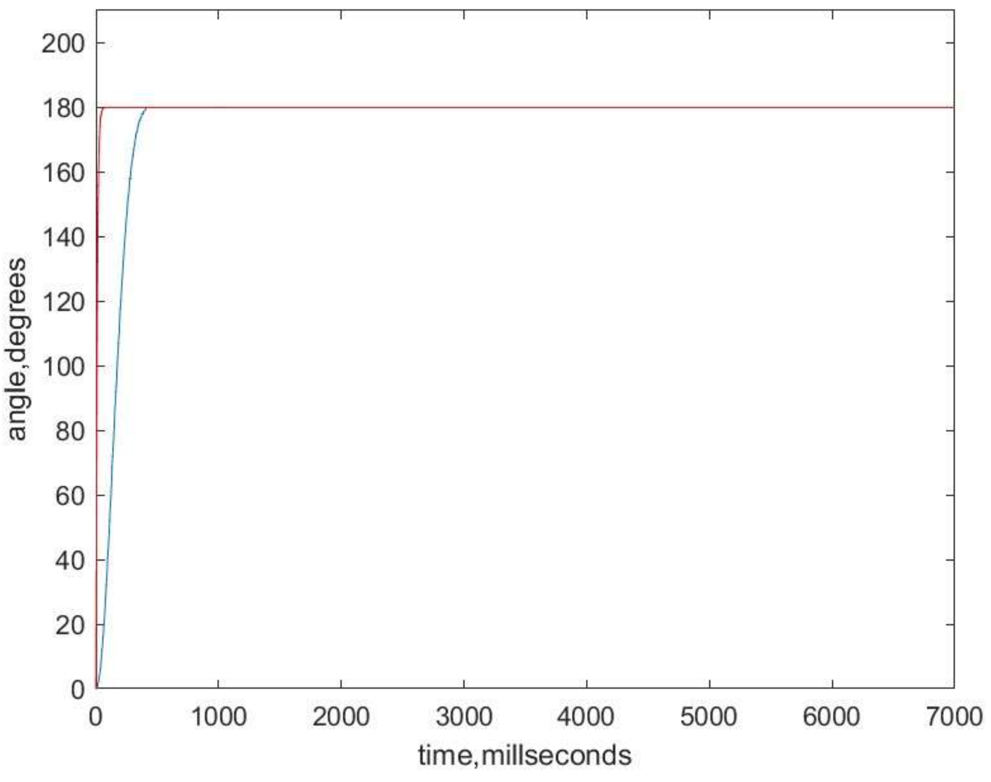
停止时间 7000
普通
快速重启
ode113

Sim(Red) vs Real

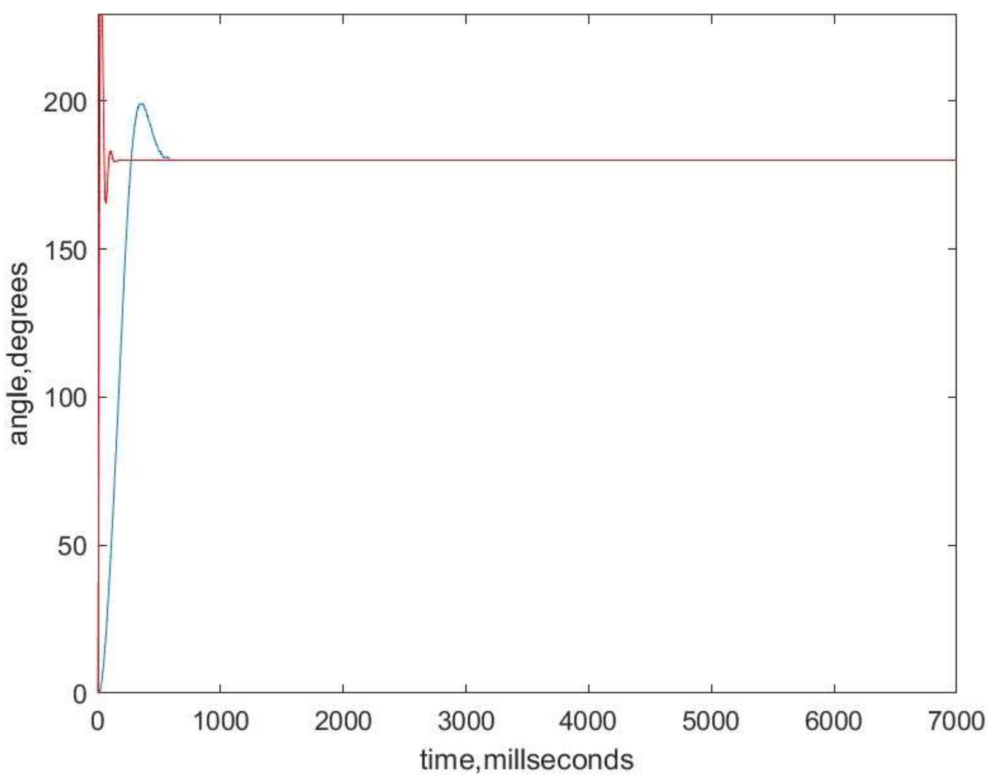


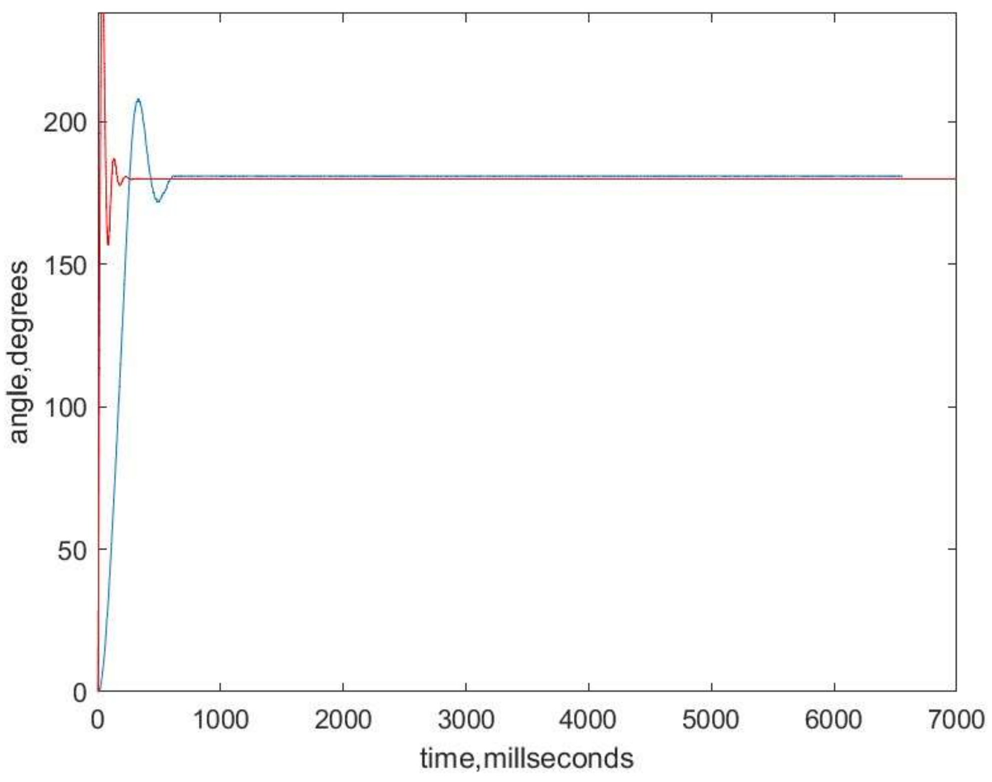


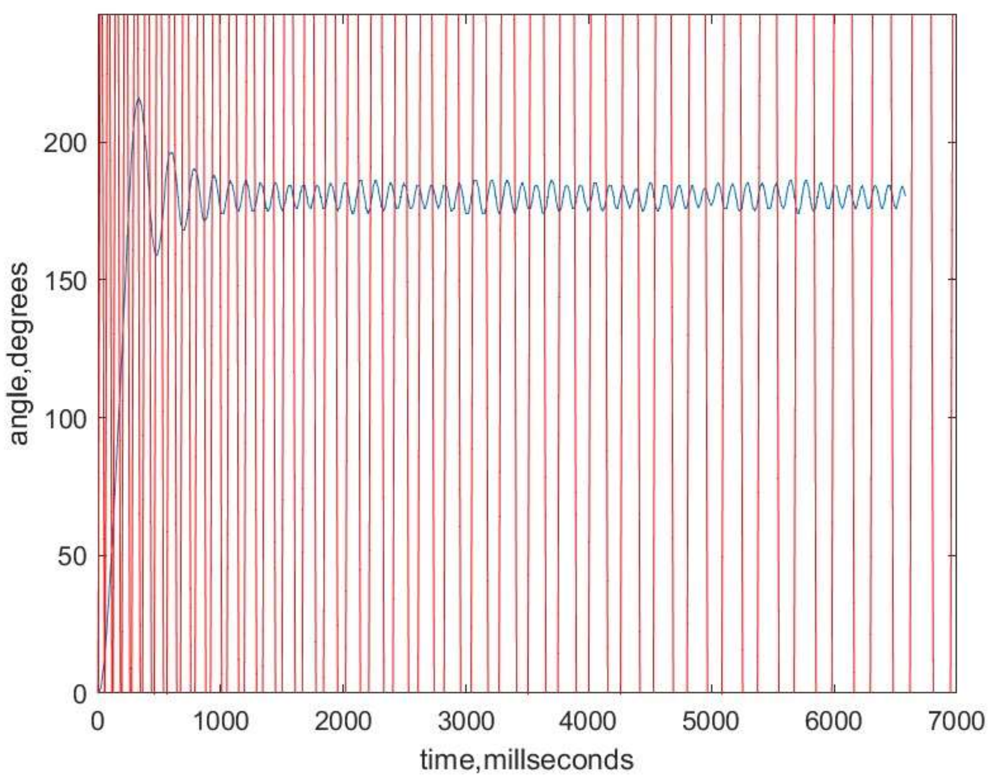
angle,degrees
time,millseconds
0
20
40
60
80
100
120
140
160
180
0
1000
2000
3000
4000
5000
6000
7000



angle,degrees
time,millseconds
0
20
40
60
80
100
120
140
160
180
200
0
1000
2000
3000
4000
5000
6000
7000


angle,degrees
0
50
100
150
200
time,millseconds
0
1000
2000
3000
4000
5000
6000
7000


angle,degrees
0
50
100
150
200
time,millseconds
0
1000
2000
3000
4000
5000
6000
7000



Conclusion:

Kp rises, rise time decrease, overshoot rise, steady error decrease.
Ki rises, rise time decrease, overshoot rise, steady error decrease, settling time lengthen.
Kd rises, rise time increase, overshoot decrease, settling time lengthen.
The motor has a huge friction.