山东大学

崇新学堂

2022－2023 学年第一学期

实 验 报 告

课程名称：EECS

实验名称：Designlab6

专 业 班 级 21 级崇新

学 生 姓 名 张原 刘浩 施政 池弋

实 验 时 间 2022.10.24

Check Yourself 1. Draw a block diagram of the controller.

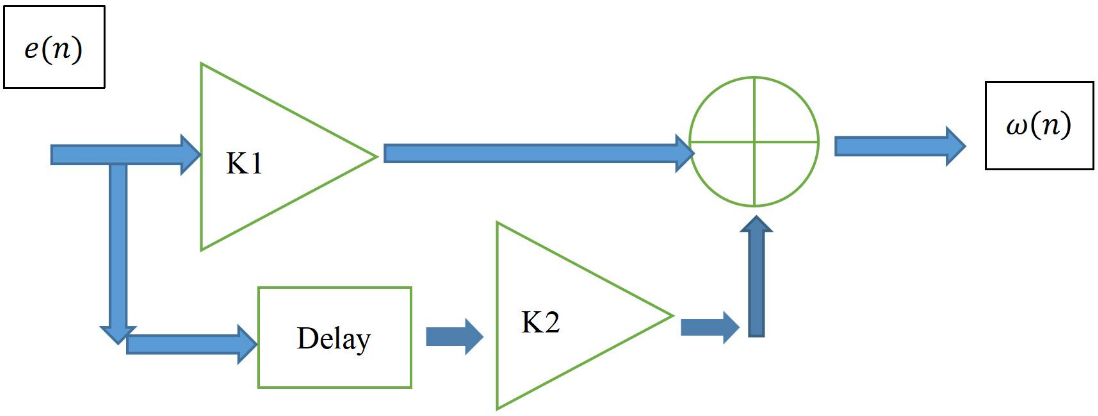


Figure 1 the controller

Step1. write designLab06Work.py

The controller just needs to multiply the signal by k1, delay it by one time step and multiply it by k2, and add the results. we're going to use sf.FeedforwardAdd to add the results together. Plant 1 and 2 are the same as in designlab5.

The whole system code is shown in the figure 2:

```
def delayPlusPropModel(k1, k2):
    T = 0.1
    V = 0.1

    # Controller:  your code here
    controller = sf.FeedforwardAdd(sf.Gain(k1),sf.Casade(sf.R(),sf.Gain(k2)))
    # The plant is like the one for the proportional controller.  Use
    # your definition from last week.
    plant1 = sf.Cascade(sf.Cascade(sf.R(),sf.Gain(T)),sf.FeedbackAdd(sf.Gain(1),sf.R()))
    plant2 = sf.Cascade(sf.Cascade(sf.R(),sf.Gain(V*T)),sf.FeedbackAdd(sf.Gain(1),sf.R()))
    # Combine the three parts
    sys = sf.FeedbackSubstract(sf.Casade(controller,sf.Casade(plant1,plant2)))
    return sys
```

Figure 2 system code

Step2. A pair of gain

According to the four difference equations, we can calculate that:

$$\frac{VT^2R^2(k_1 + k_2R)}{(1 - R)^2 + VT^2k_1R^2 + VT^2k_2R^3}$$

In the function of the whole system, R is replaced by 1/Z:

$$\frac{VT^2(k_1Z+k_2)}{Z^3-2Z^2+(1+VT^2k_1)Z+VT^2k_2}$$

Let the denominator be equal to 0, and let's fix k1:

$$\mathrm{Z^3-2Z^2+(1+VT^2k_1)Z+VT^2k_2=0}$$

The gain pair is obtained using the function as follows:

k1	k2	magnitude of dominant pole
10	-9.8	0.99
30	-29.76	0.98
100	-97.34	0.946
300	-271.68	0.772

Figure 3 result

Step3.DelayPlusPropBrainSkeleton

We think that in WALLFOLLOWER, the state should store our expected distance to calculate the error.Let the machine output io.Action at the end.

The code is shown below:

```
desiredRight = 0.5
forwardVelocity = 0.1
k1 = 100
k2 = -97
# No additional delay
class Sensor(sm.SM): #输入io 输出到墙的距离
    def getNextValues(self, state, inp):
        v = sonarDist.getDistanceRight(inp.sonars)
        print 'Dist from robot center to wall on right', v
        return (state, v)

# inp is the distance to the right
class WallFollower(sm.SM):
    startState = [desiredRight]
    def getNextValues(self, state, inp):
        e1 = desiredRight - inp
        e2 = desiredRight - state[0]
        w = k1*e1 +k2*e2
        return ([inp],io.Action(fvel = forwardVelocity,rvel = w))
```

Figure 4 code of DelayPlusPropBrainSkeleton.py

Step4.testing performance in simulation

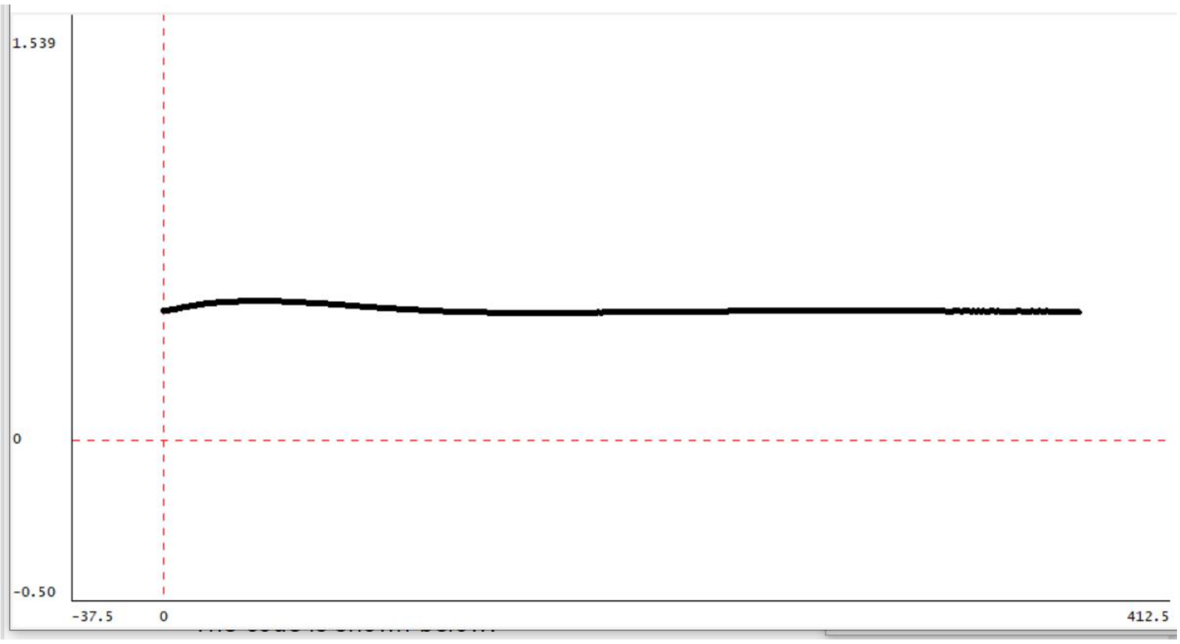


Figure 5 k1=10,k2=-9.8

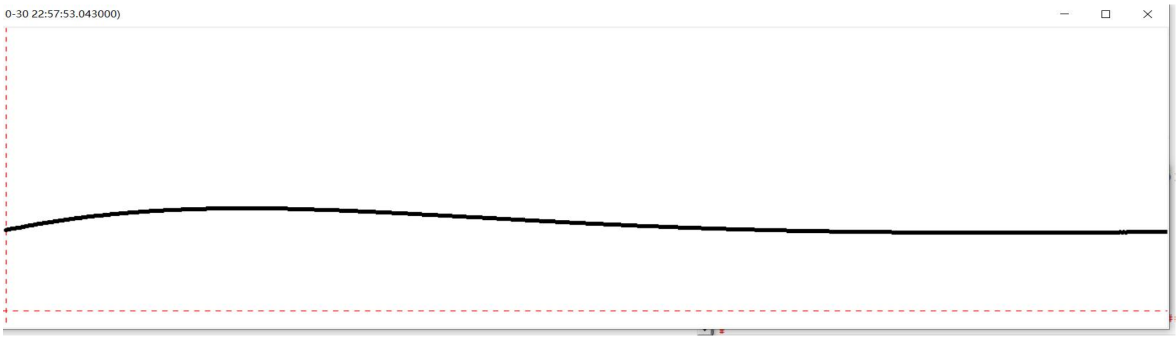


Figure 6 k1=30,k2=-29.76

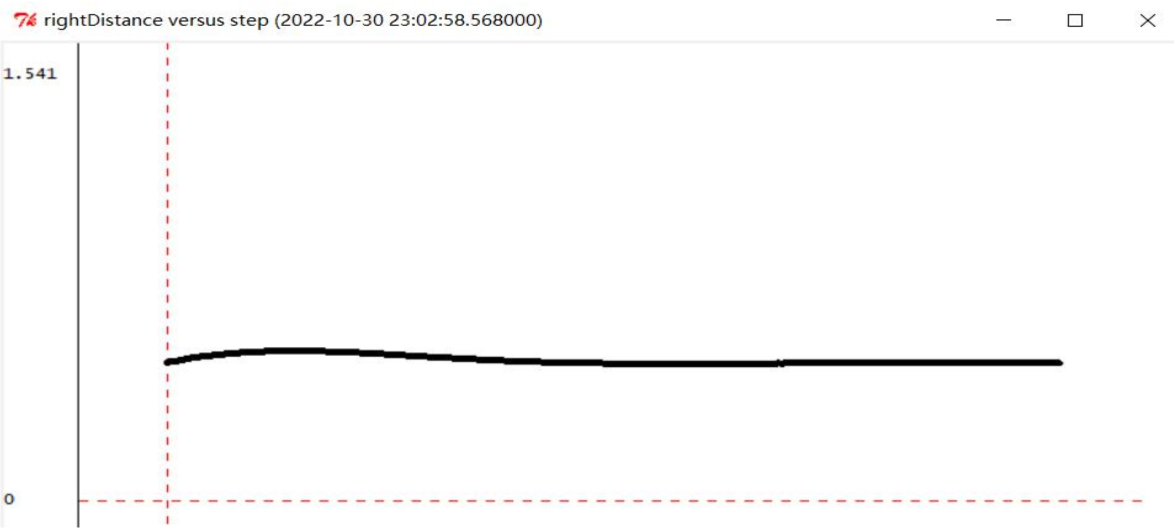


Figure 7 k1=100,k2=-97.34

Figure 8 k1=300,k2=-271.68

Check Yourself 3. **Which of the four gain pairs work best in simulation?**

From the simulation experiment, k1=300,k2=-271.68 is the best gain pair. From the simulation experiment, the convergence speed of the second group of data is the slowest.The smaller the main pole, the faster the convergence speed, and the faster the car can adjust to a stable position

Step5.testing performance

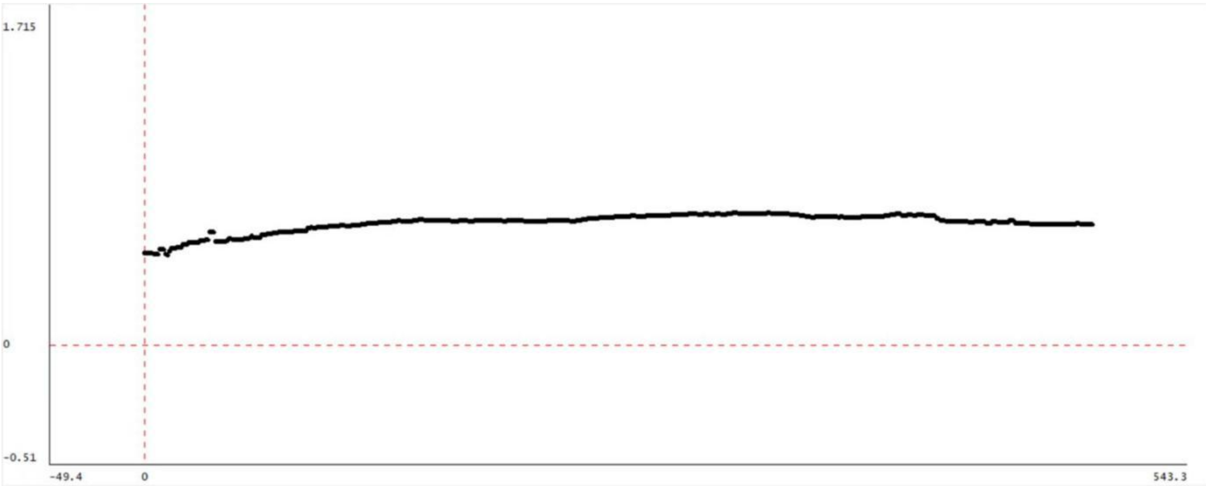


Figure 9 k1=10,k2=-9.8

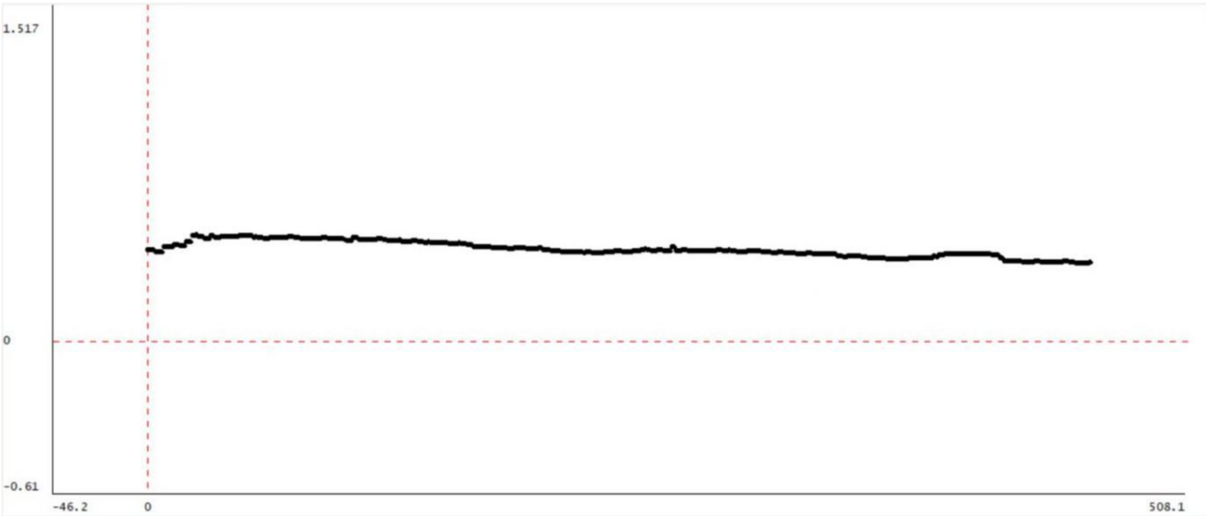


Figure 10 k1=30,k2=-29.76

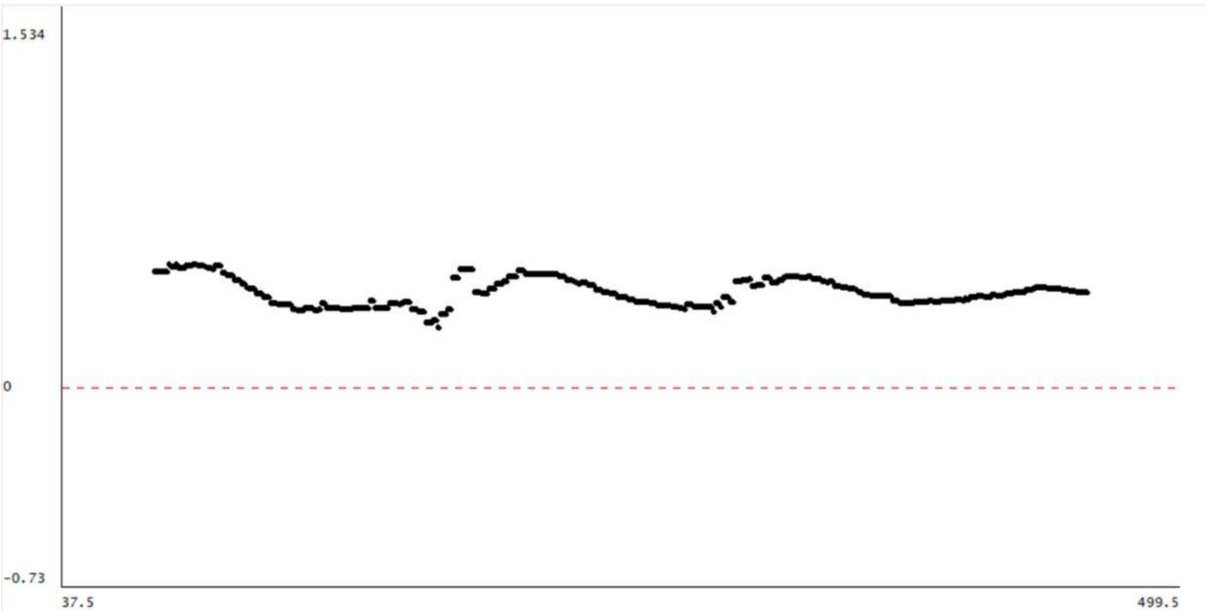


Figure 11 k1=100,k2=-97.34

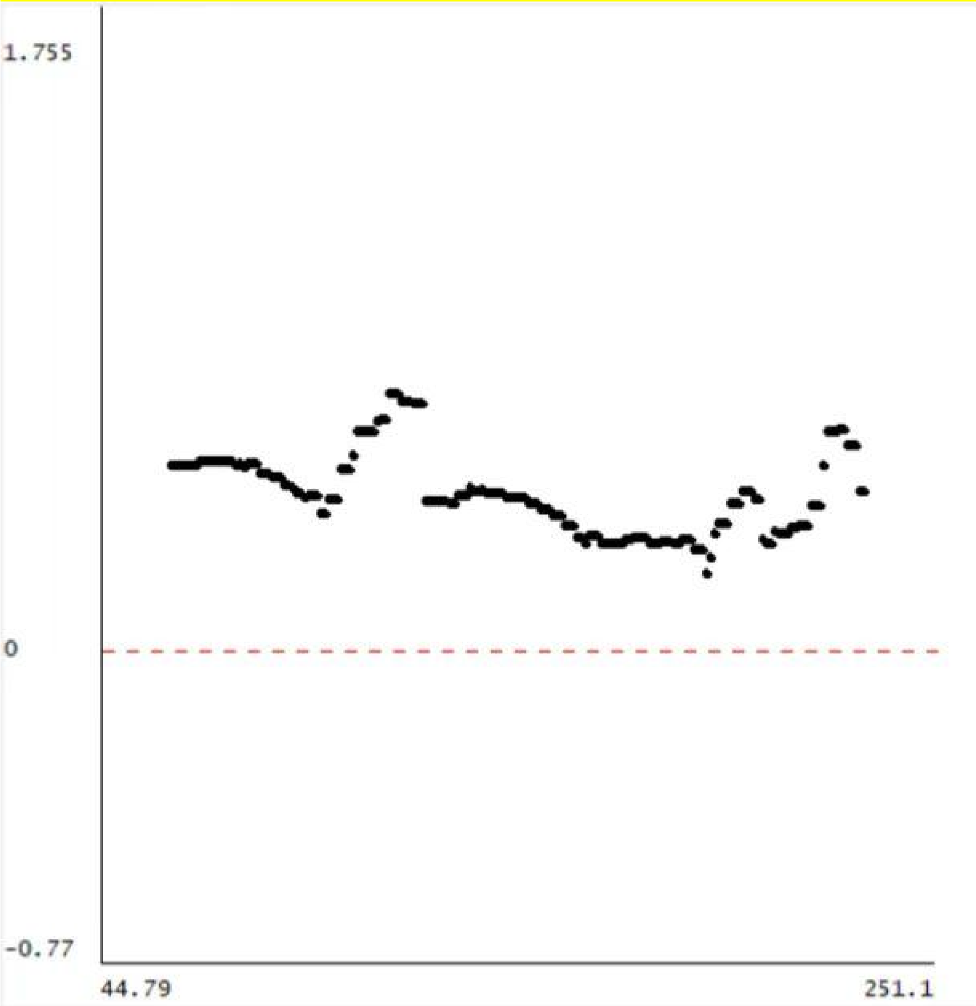


Figure 12 k1=300,k2=-271.68

Check Yourself 4.

Which of the four gain pairs work best on a robot?

From the experiment, k1=30,k2=-29.76 is the best gain pair.

Are the best gains the same as in simulation?

NO,It's difference from simulation.

Which gains cause bad behavior?

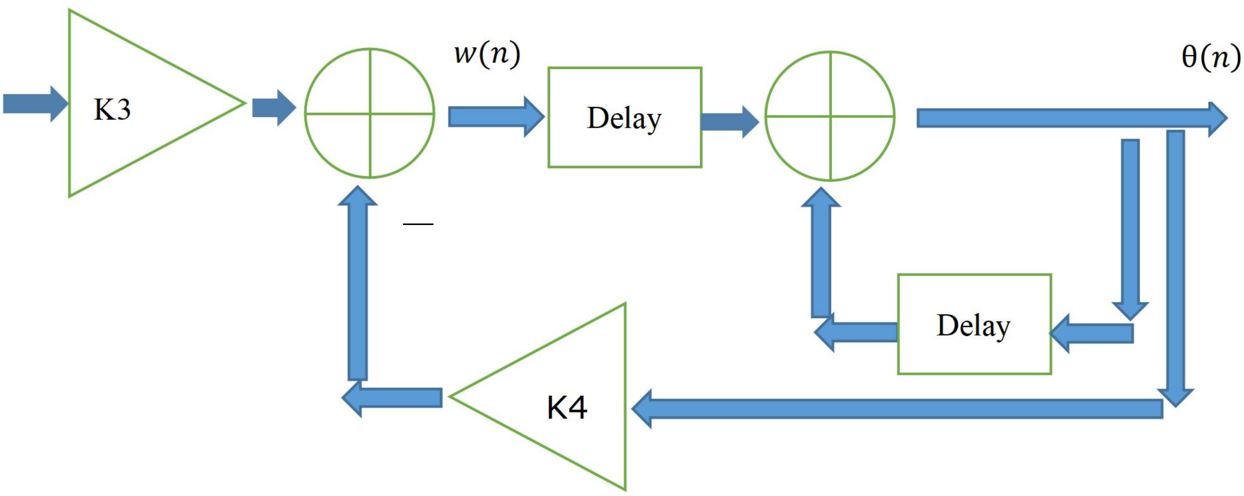
It happens to be the best result of the simulation.In actual control, too much vibration can easily lead to hitting the wall.The picture below shows the car hitting the wall in this case.

Figure 13 k1=300,k2=-271.68,car hit the wall

Checkoff 1. **Wk.6.1.2: Show a staff member plots for the simulated and real robot runs, and discuss their relationship. How is the robot's behavior related to the magnitude of the dominant pole, for each of the gain pairs? Explain how you chose the starting state of your controller.**

The larger the absolute value of the gain pair is, the easier the car is to oscillate in the experiment. Although it has a faster convergence speed, it is easy to overrotate, resulting in the collision with the wall.

Step6. **angle-plus-proportional control**



Step7.The function and result of angle-plus-proportional control

1According to the four difference equations, we can calculate that:

$$\frac{\mathrm{VT^2R^2k_3}}{(1-\mathrm{R})^2+\mathrm{TRk_4}-\mathrm{TR^2k_4}+\mathrm{VT^2k_3R^2}}$$

In the function of the whole system, R is replaced by 1/Z:

$$\frac{VT^2k_3}{Z^2+(Tk_4-2)Z+1+VT^2k_3-Tk_4}$$

Let the denominator be equal to 0, and let's fix k3:

$$\mathrm{Z^2+(Tk_4-2)Z+1+VT^2k_3-Tk_4=0}$$

The gain pair is obtained using the function as follows:

k3	k4	magnitude of dominant pole
1	0.632	0.968
3	1.091	0.945
10	2.0	0.899
30	3.46	0.827

Figure 14 result of angle-plus- proportional control

Step8.code

Just change e2 and the angular velocity, and the code is as follows:

```
desiredRight = 0.5
forwardVelocity = 0.1
k3 = 1
k4 = 0.632

# No additional delay.
# Output is a sequence of (distance, angle) pairs
class Sensor(sm.SM):
   def getNextValues(self, state, inp):
       v = sonarDist.getDistanceRightAndAngle(inp.sonars)
       print 'Dist from robot center to wall on right', v[0]
       if not v[1]:
           print '****** Angle reading not valid ******'
       return (state, v)


# inp is a tuple (distanceRight, angle)
class WallFollower(sm.SM):
    startState = 'False'
    def getNextValues(self, state, inp):
        (distanceRight, angle) = inp
        e1 = desiredRight - distanceRight
        e2 = k4*angle
        w = k3*e1 - k4*e2
        return ('False',io.Action(fvel = forwardVelocity,rvel = w))
```

Figure 15 code

Step9.

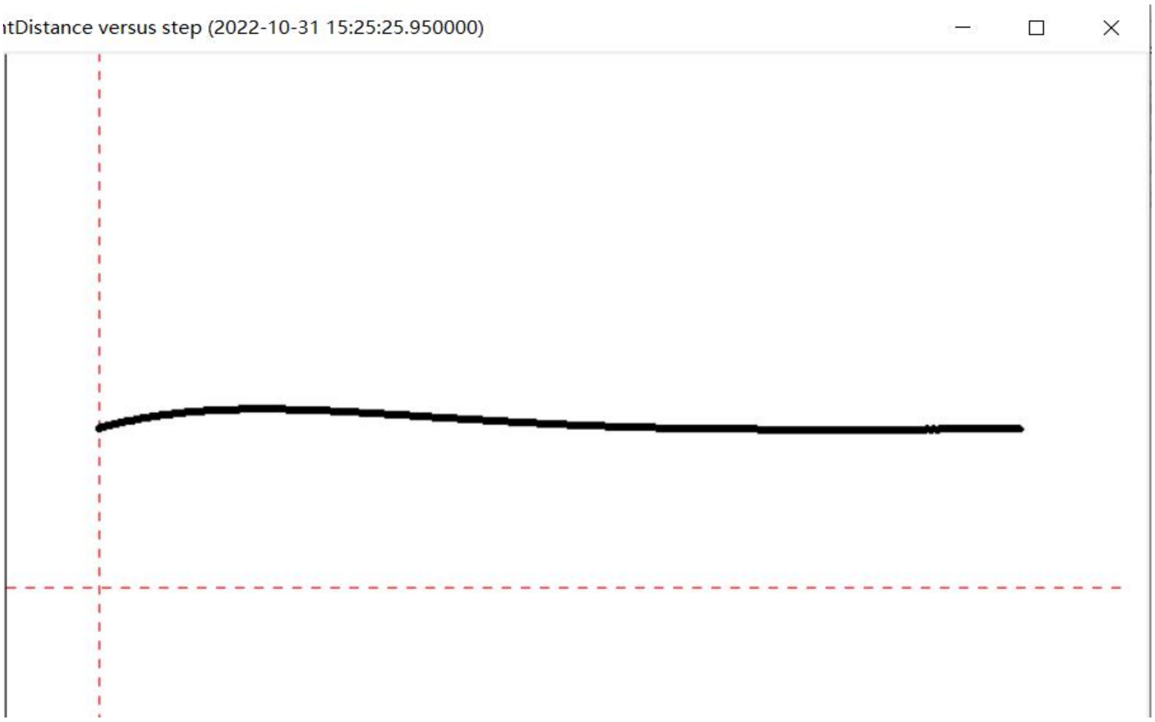


Figure 16 k3=1,k4=0.632

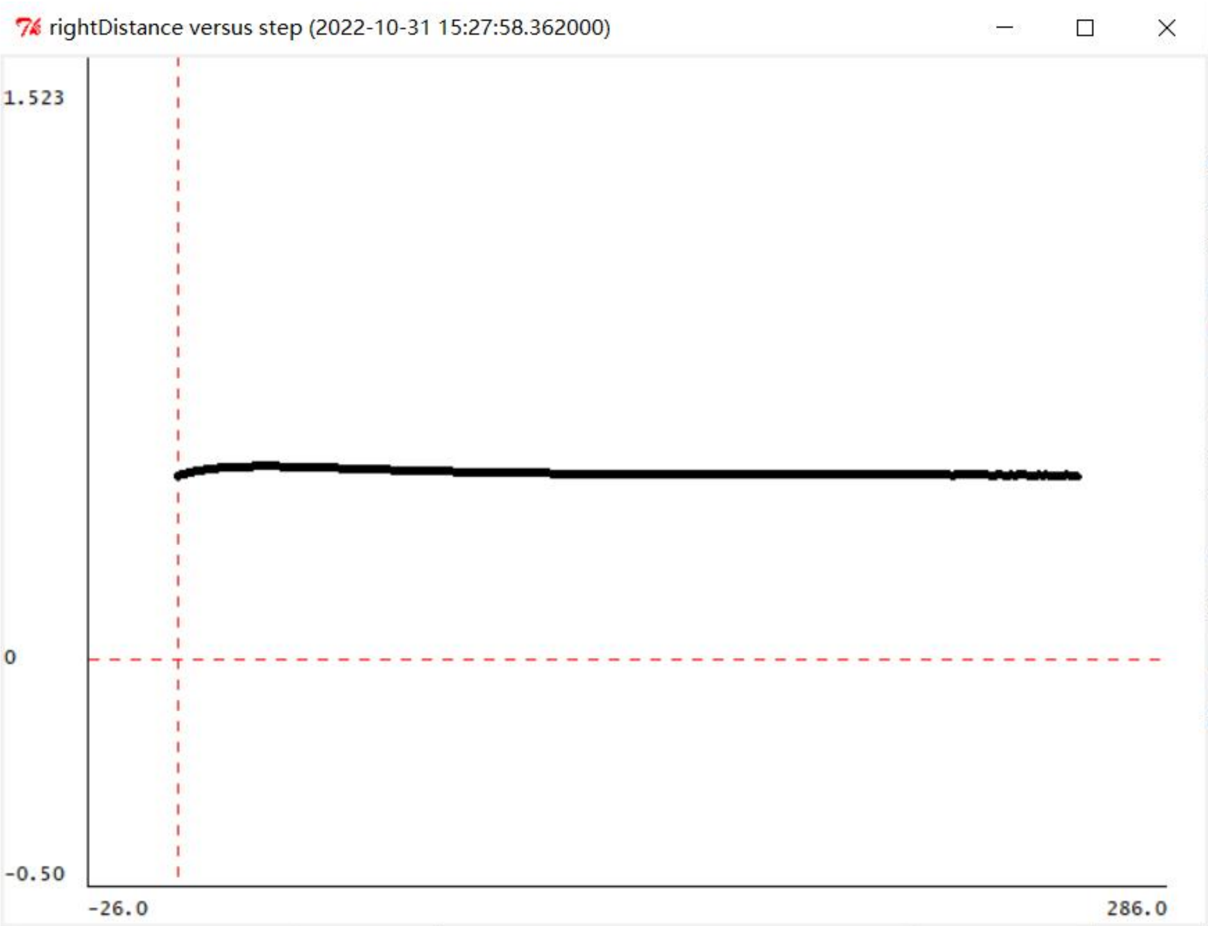


Figure 17 k3=3,k4=1.091

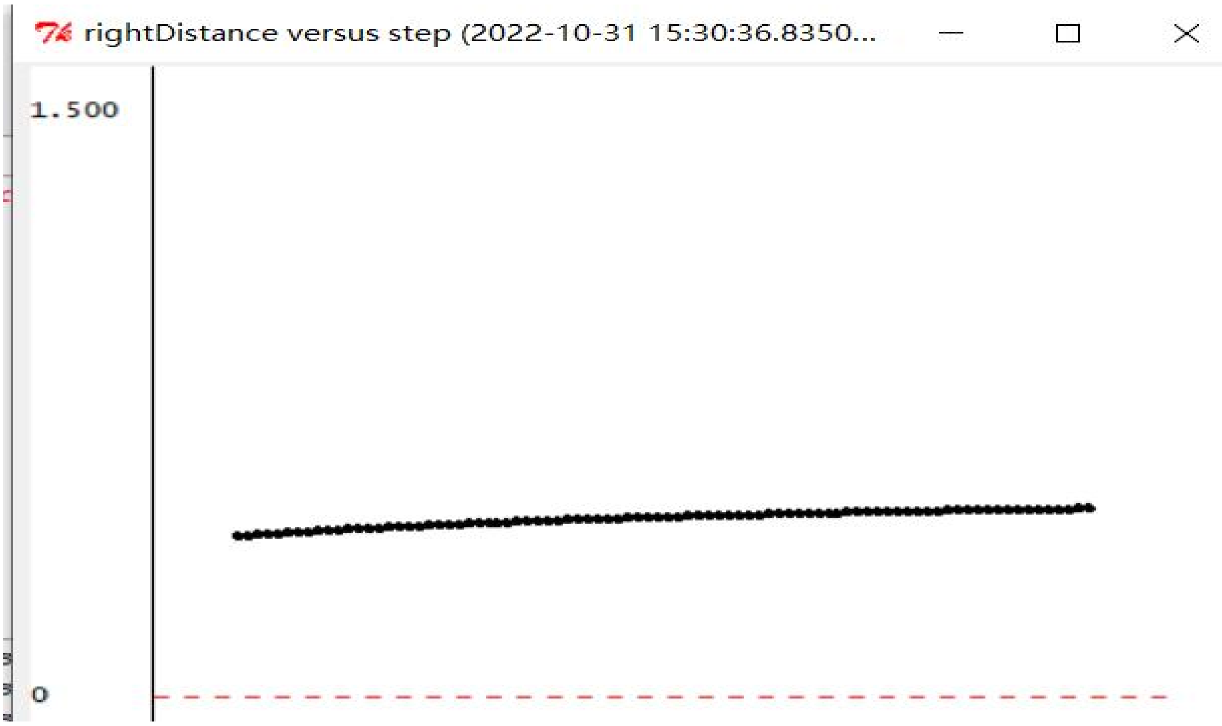


Figure 18 k3=10,k4=2.0

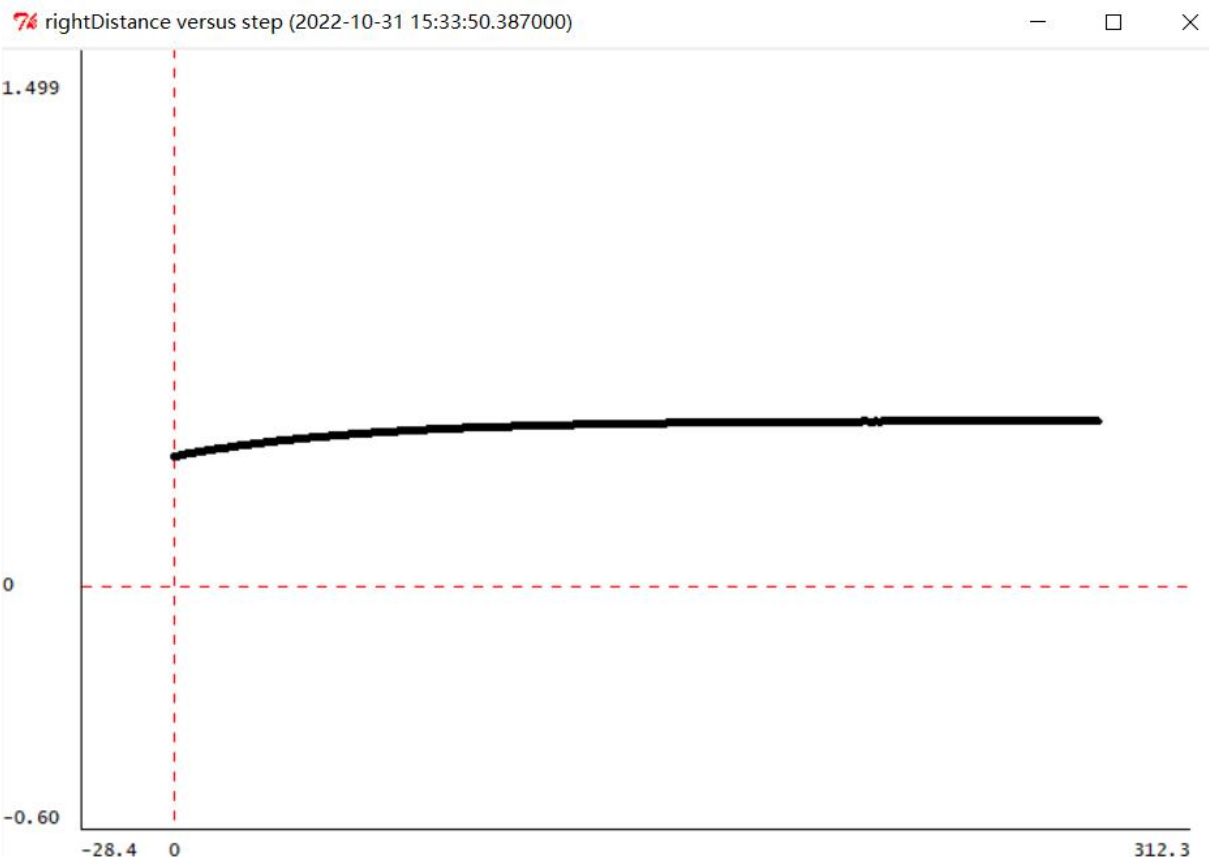


Figure 19 k3=30,k4=3.46

Check Yourself 5.

Which of the four gain pairs work best in simulation?

k3 =3 k4 =1.091

Which gains cause bad behavior?

In simulation experiments, each gain pair looks good

Step10.

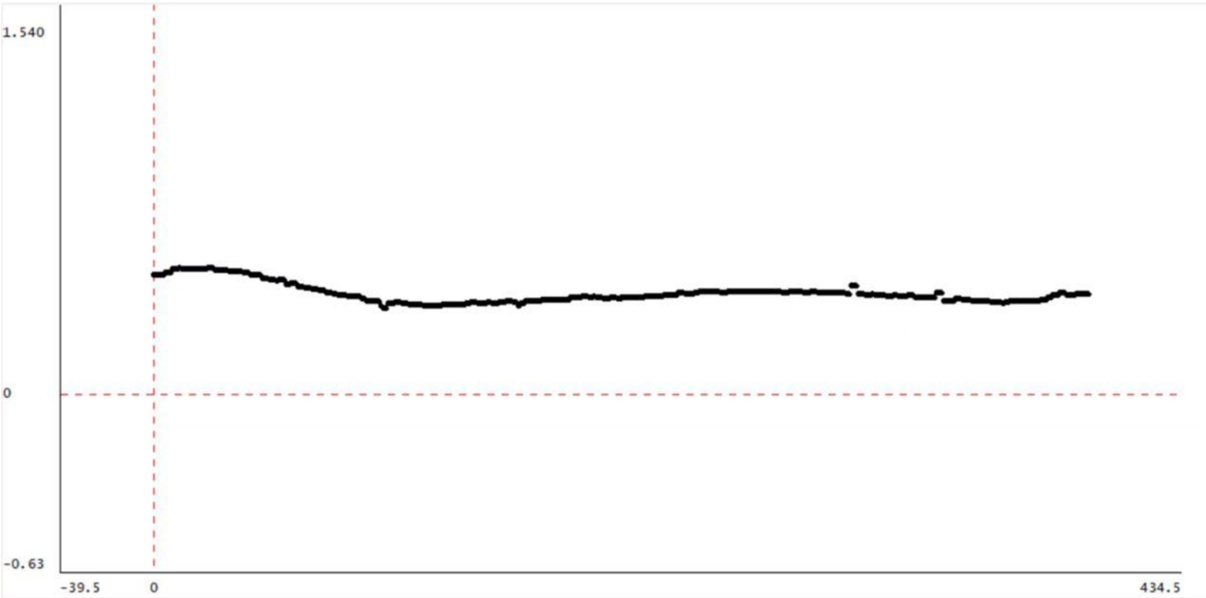


Figure 20 k3=1,k4=0.632

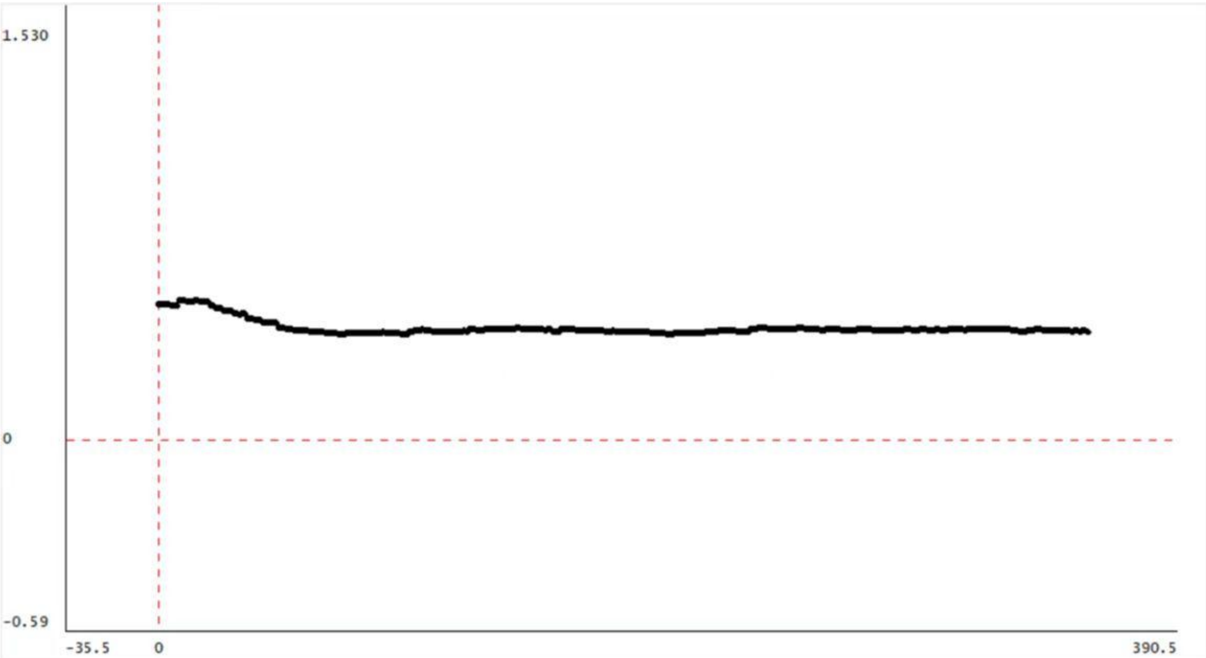


Figure 21 k3=3,k4=1.091

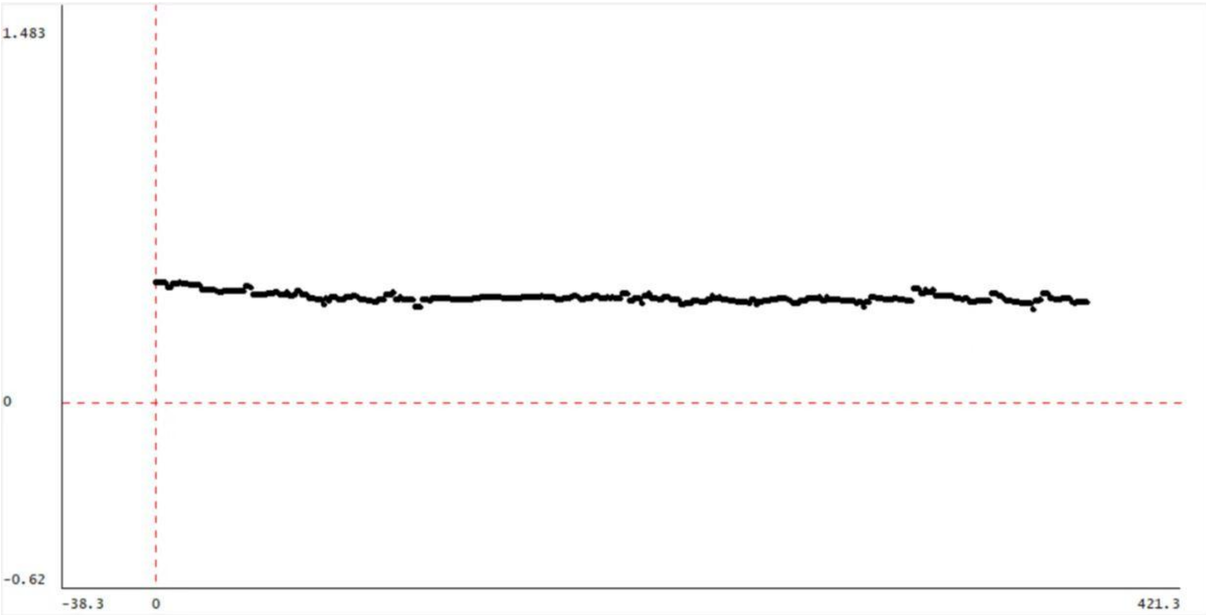


Figure 22 k3=10,k4=2.0

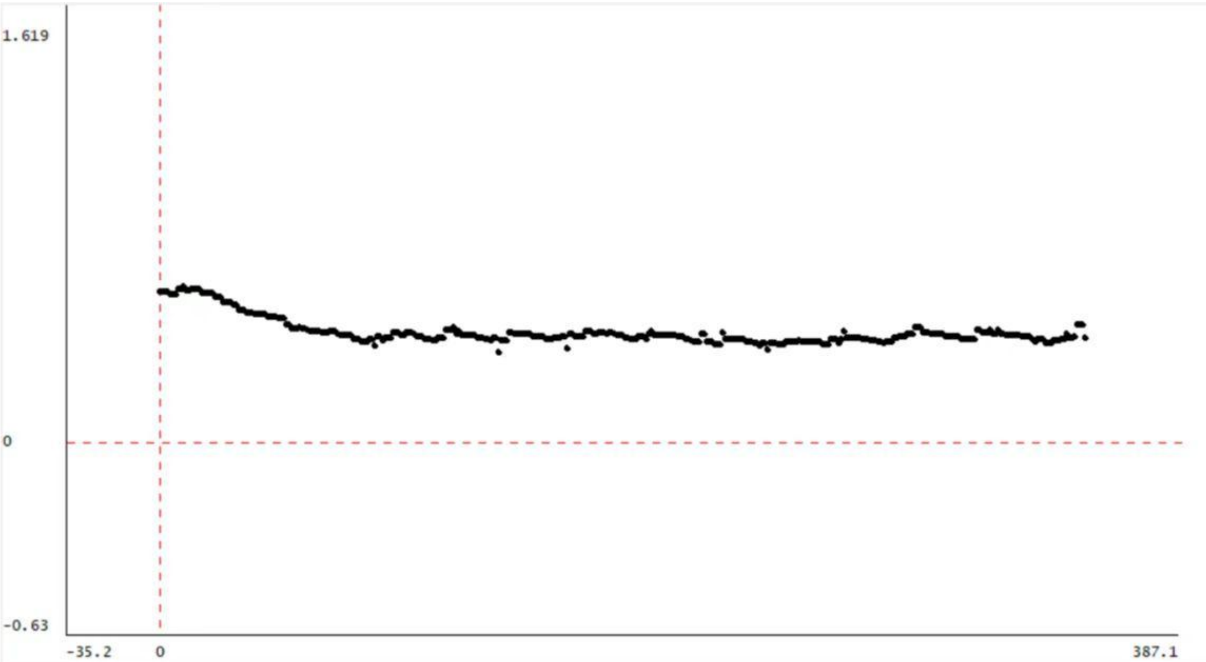


Figure 23 k3=30,k4=3.46

Check Yourself 5.

Which of the four gain pairs work best on a robot?

From the experiment, k3=3,k4=1.091 is the best gain pair.

Are the best gains the same as in simulation?

Yes,both the simulation results and the actual results show that the second set of gain pairs is the optimal gain pair

Which gains cause bad behavior?

The fourth set of gain-pair-determined car actions seemed to oscillate and not converge, and did not perform as well in the real car experiment as in the simulation.

Checkoff 2.

Wk.6.1.4: Show a staff member plots for the simulated and real robot runs, and discuss their relationship. How is the robot's behavior related to the magnitude of the dominant pole, for each of the gain pairs?

The smaller the main pole, the faster the convergence speed, and the faster the car can adjust to a stable position

Which controller (delay-plus-proportional or angle plus proportional) performs better? Explain why.

According to the experiment, the angle-plus-proportional is better. angle-plus-proportional reduces the delay and reduces the order of the denominator of the system function, reduces the clutter interference, and the effect is better.

Summary

1.In this experiment, we improved the system of the previous two experiments, so that the car can walk along the wall more smoothly.

2.In the first simulation, the best simulation result appeared but the actual effect was the worst. After discussion, we believed that this situation was due to the fact that the real physical environment could not synchronize the transmission of the current value and the delay value.

3.In the real car, it should be considered that the real physical system is different from the simulation. In theory, the smaller the pole is, the easier it is to converge, but there is a certain delay in the actual system, which will cause oscillation.