山东大学

崇新学堂

2025－2026 学年第一学期

实 验 报 告

课程名称：　电子信息工程导论

实验名称：　Staggering Proportions

专　业　班　级　崇新学堂

学　生　姓　名　高子轩，钱竹玉，吕思洁，徐亚骐

实　验　时　间　2025 年 10 月 19 日

Introduction

This week, we study a feedback system based on sonar detectors in a two-dimensional situation. By modeling a difference equation, we aim to control the robot to move parallel to the wall.

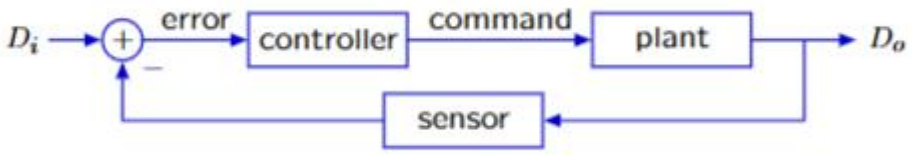

Figure 1 The basic structure of the system

Proportional wall-follower

Our approach is to give it a fixed forward speed V and adjust the rotational speed ω to make it directly proportional to the "error", that is：

$$\omega[n] = k(d_i[n] - d_o[n])$$

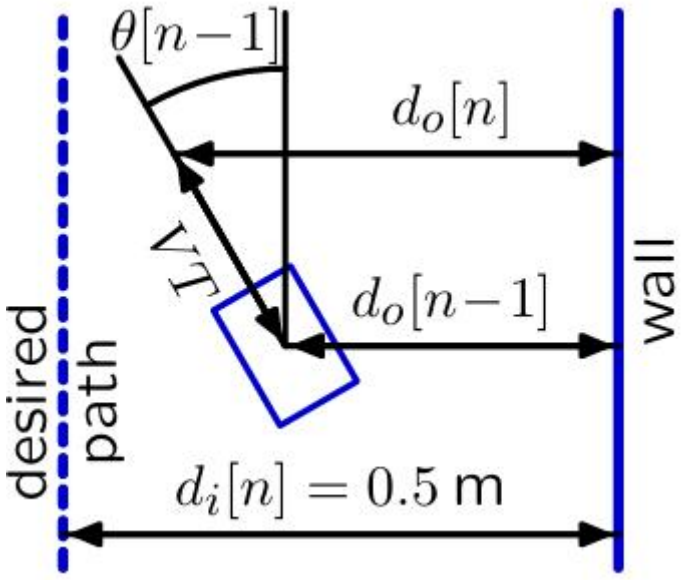

Figure 2 Demonstration diagram

Check Yourself 1:

Explanation：The WallFollower receives the output from the Sensor, and the Sensor outputs the actual distance $d_o[n]$. $e[n] = d_i[n] - d_o[n]$, so the input type of the WallFollower is float. The output is the angular velocity $\omega[n]$, which is also of the float type

If $d_o[n]$ is too small (the robot is too close to the wall), then $e[n] = d_i[n] - d_o[n] > 0$. At this point, the robot should be away from the wall. Because when the angular velocity $\omega[n]$ is positive, the robot turns left, so positive error → positive $\omega[n]$ → turn left. Therefore, in $\omega[n] = ke[n]$, k should be a positive number.

Conclusion：

Input：float output：io.Rotation k：positive

Checkoff1: Explain how gain k affects the wall follower's behavior

K's function to the car

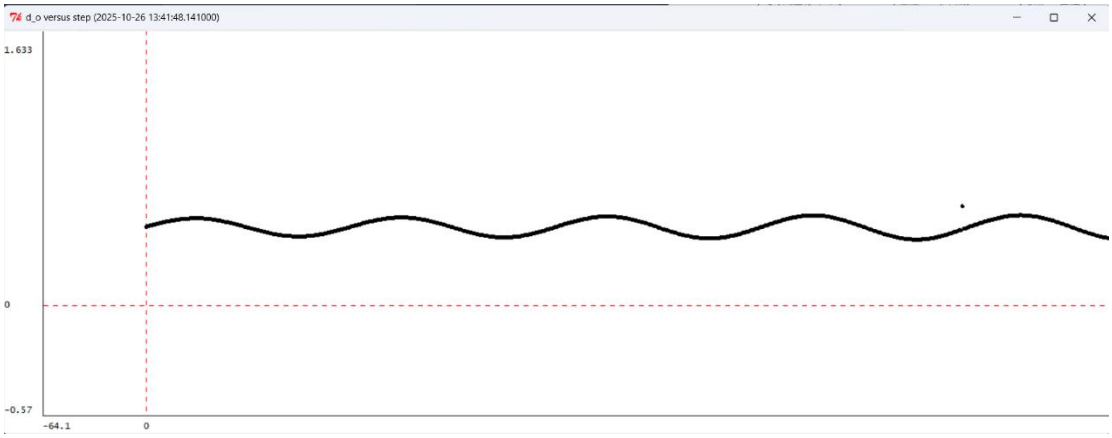

Figure 3 plot when

k=15

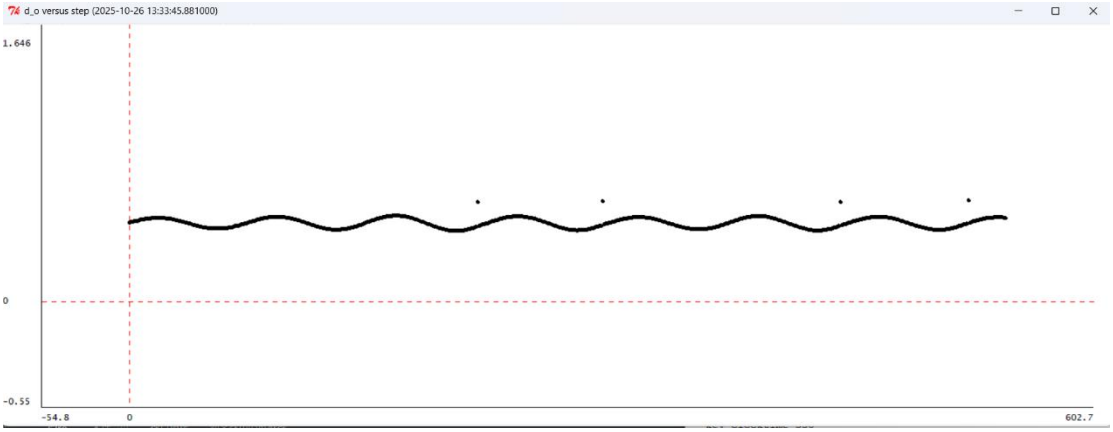

Figure 4 plot when

k=5

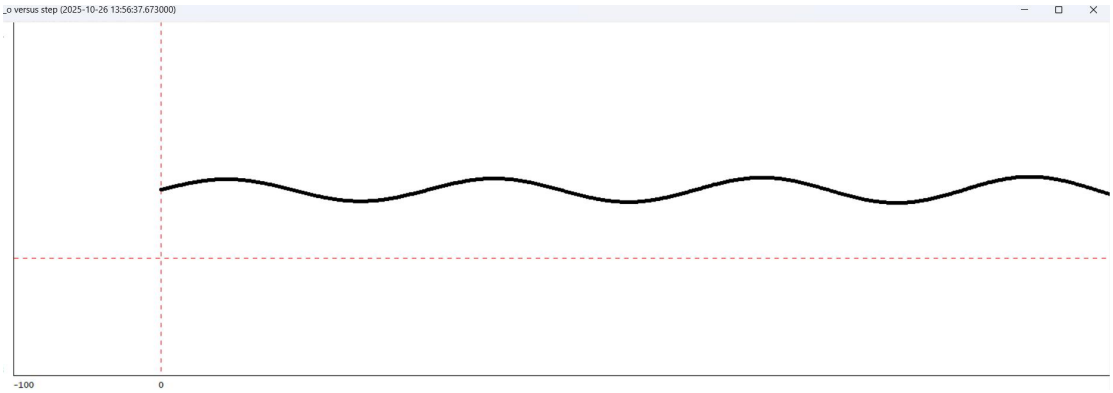

Figure 5 plot when k=2.5

As shown in the figure above, when k increases, the oscillation becomes faster with a shorter period, while the oscillation weakens and converges to $d_i[n]$ with little noticeable effect.

We haven't identified the fastest-converging k value. The optimal k value currently appears to be 5.

Step 3: Constructing the Controller, Plant1, Plant2

Relationship Equations

The Controller

$$\omega[n] = k(di[n] - do[n])$$

Plant1

$$\theta[n] = \theta[n-1] + T\omega[n-1]$$

Plant2

$$d_o[n] = do[n-1] + TV\theta[n-1]$$

Explanation for Plant2

The original equation would be

$$d_o[n] = do[n-1] + TVsin(\theta[n-1])$$

We made an approximation $\theta = sin\theta$ when θ is very small. Convert the sine relationship to a linear relationship. Then we obtain the answer.

Comparison with lab04

First, the control in Lab 04 is a first-order system, which directly affects the change of distance; while the control in Lab 05 is a second-order system, which requires two summations to affect the final output.

Secondly, due to this second-order characteristic, Lab 05 is more sensitive to the k value, which is prone to continuous and difficult-to-decay oscillations, while Lab 04 can relatively easily find a k value that can achieve stable convergence.

Step5 Convert difference equations into operator (R) equations

Controller model

$$\omega = k(D_i - D_o)^{\leftarrow}$$

Plant 1

$$\theta = R\theta + TR\omega$$

Plant 2

$$D_o - RD_o = vTR\sin\theta \approx vTR\theta$$

By combining the three equations above, we can establish a relationship between D_i and D_o:

$$H = \frac{D_o}{D_i} = \frac{kvT^2R^2}{kvT^2R^2 + (1-R)^2}$$

Step 6 Find an algebraic expression for the poles p of the system

Assuming that $z = \frac{1}{R}$,we get $H = \frac{kvT^2}{kvT^2+(z-1)^2}$

Then, we can set the denominator to zero to solve the condition

$$kvT^2 + (z-1)^2 = 0$$

This leads to the solution:

$$z = 1 \pm iT\sqrt{kv}$$

$p = \frac{1}{z}$ so that is to say

$$p = \frac{1 \pm iT\sqrt{kv}}{1 + kvT^2}$$

Step 7 find the information about the response of the

system

For the case of $p = \frac{1 \pm iT\sqrt{kv}}{1+kvT^2}$, when $k = 1, T = 0.1, v = 0.1$,

The corresponding system response is shown in the figure below:

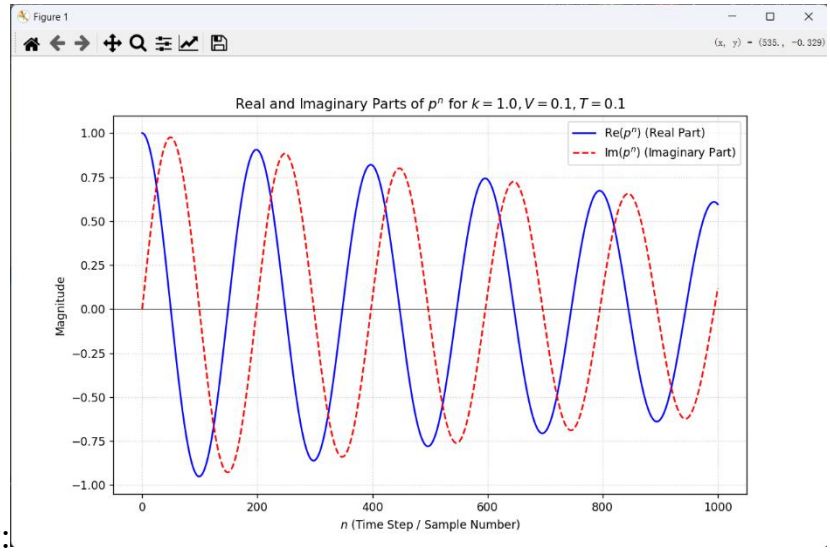

Figure 6 Real and imaginary parts of pole

Since the pole magnitude $p = 0.999$ is stable but extremely close to the unit circle, the oscillations will decay very slowly, resulting in a prolonged transient period before convergence to the desired steady state.

Checkoff 2

The calculation results show that at this time, the pole of the system is located at $p = 0.999 + 0.0316i$. This position reveals several pieces of information. Firstly, the modulus of the pole $|p| \approx 0.999$ is slightly less than 1. However, since it is so close to the unit circle, the decay will be extremely slow, taking a long time to come to a complete stop.

The influence of pole positions on system behavior::

Since the amplitude of the poles is less than 1, the system is stable and will eventually converge to a steady state.

However, because the poles are very close to the unit circle, the convergence speed is extremely slow, the damping ratio is small, and it takes a big time step for the decaying oscillation phenomenon to become more obvious.

Comparison with the results of Checkoff 1 experiment:

Similarities :

Both systems exhibit significant oscillation, with the oscillation gradually diminishing as the time step increases.

Testing of k values reveals that the oscillation period decreases as k increases in both systems.

Differences:

The theoretical model predicted an extremely slow process, whereas the actual system converged much faster.

Differences interpretation:

The approximation $sin\theta \approx \theta$ used may introduce errors when the actual steering angle is large.

Software model

We need to use the SystemFunction class to construct a complete

system block diagram to achieve the purpose of automatically handling a large number of algebraic operations through software editing

Check Yourself 2: Use gains,delays to build system such as Controller

For Controller:

$$\omega[n] = ke[n]$$

According to this formula, the input of the Controller is e[n], and the output w[n] should be k times of it, so we need a gain with a value of k.

The diagram is as follow:

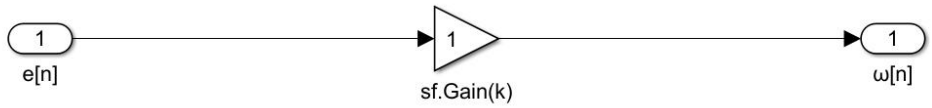

Figure 7 the diagram for Controller

For Plant1:

$$\theta[n] - \theta[n-1] = T\omega[n-1]$$

According to this formula, the input ω [n] passes through a gain T and a unit delay, and then the resulting structure is added to the delayed output to obtain the output θ [n]. Positive feedback is constructed by applying gain and delay to the input before providing feedback.

The diagram is as

follow:

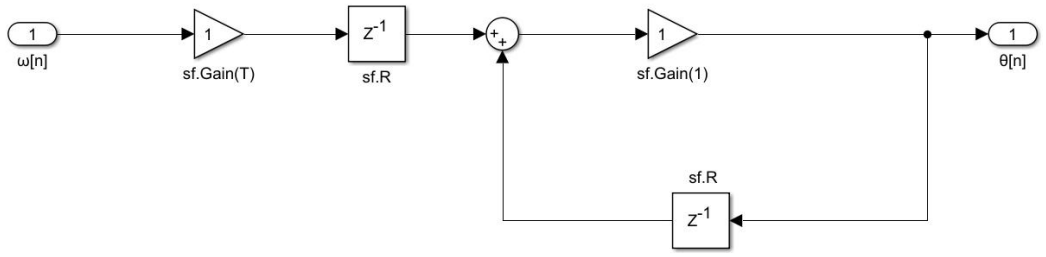

Figure 8 the diagram for Plant1

For Plant2:

$$d_o[n] - d_o[n-1] = Tv\theta[n-1]$$

According to this formula, the structure is largely similar to Plant1, and the system diagram can be obtained by simply adjusting the amplification factor

The diagram is as follow:

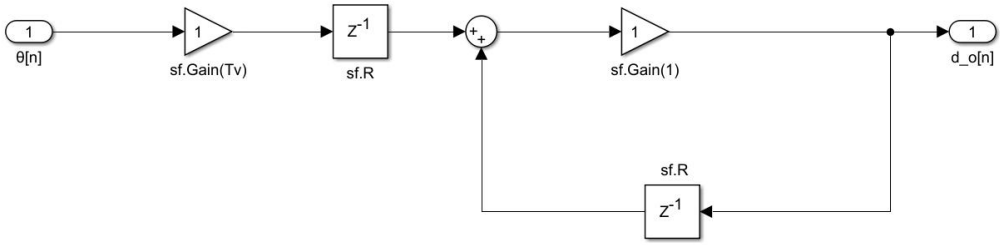

Figure 9 the diagram for Plant2

Step8: Fill the code for controller, plant1, and plant2 in designLab05Work.py

For Controller:

We simply need to apply a gain to the input to achieve the desired output.

Here is the key code

```
def controller(k):
   return sf.Gain(k)
```

For Plant1:

We should apply a gain of T and then apply a unit delay. Then, use sf.FeedbackAdd() with the forward path as a unit delay and the feedback path as a gain of 1.

Here is the key code

```
def plant1(T):
   FeedbackAdd = sf.FeedbackAdd(sf.Gain(1), sf.R())
   return sf.Cascade(sf.Gain(T),sf.Cascade(sf.R(), FeedbackAdd))
```

For Plant2:

The logic is similar to the Plant1, we just need to replace the T with T*V.

Here is the key code

```
def plant2(T, V):
   FeedbackAdd = sf.FeedbackAdd(sf.Gain(1), sf.R())
   return sf.Cascade(sf.Gain(T*V),sf.Cascade(sf.R(), FeedbackAdd))
```

Step 9: Compose them into a single System Function in procedure wallFollowerModel(k, T, V)

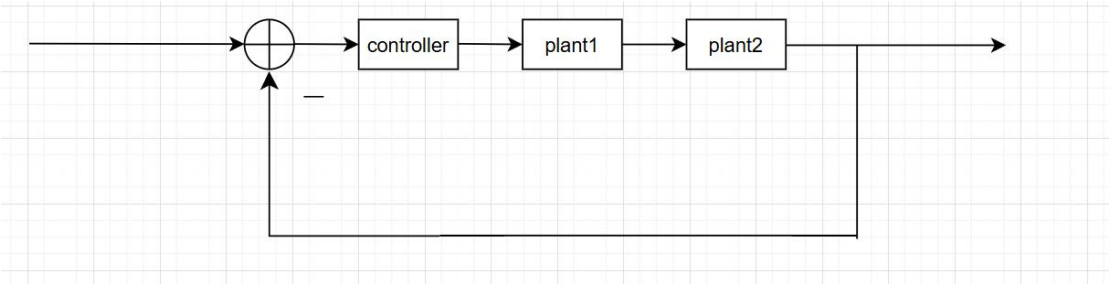

Figure 10 the whole system diagram

Code logic:

Since our input is $d_i[n]$, the system's input should be $e[n]$ as shown in the diagram. By feeding $d_o[n]$ back into the input through negative feedback, we can obtain this system.

Here is the key code

```
def wallFollowerModel(k, T, V):
   h_controller = controller(k)
   h_plant1 = plant1(T)
   h_plant2 = plant2(T, V)

   h_plant_cascade = sf.Cascade(h_plant1, h_plant2)
   h_open = sf.Cascade(h_controller, h_plant_cascade)
   return sf.FeedbackSubtract(h_open, sf.Gain(1))
```

Step 10: determine the period in Checkoff1

Code logic

Construct a mathematical model that represents the entire robot control system. Then, use the dominantPole method to find the dominant pole of the model. Determine whether the system will oscillate by checking if the imaginary part of the pole exists. If it oscillates, calculate the time required to complete one full oscillation based on the angle of the pole in the complex plane.

Here is the key code

```
def calculate_period(k_value, T_value=0.1, V_value=0.1):

   h_model = wallFollowerModel(k_value, T_value, V_value)
   p = h_model.dominantPole()

   if abs(p.imag) < 1e-9:
      return "Not Oscillating"

   phi = math.atan2(p.imag, p.real)
   period_samples = 2 * math.pi / abs(phi)
   period_time = period_samples * T_value

   return period_time
```

k	***period***
25	***4.00673267689***
15	***5.15574851168***
5	***8.90055579151***
2.5	***12.5768356177***

Table 1 k and period in step 10

Checkoff 3: similarities and differences between Checkoff1 and Step10

Similarities: The decreasing trend of the period with increasing gain k is consistent in both the model and simulation.

Differences: Due to the assumptions of small-angle linearization and instantaneous sensor response, these practical factors introduce additional damping and phase lag, resulting in differences.