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： Homework 2 Heads Up

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 28 日

Step 1

Solving steps:

With the input $e[n]$,we need to get $v_s[n]$.

Voltage $v_s[n]$ is proportional to the angle between the light's angular position and that of the head

$$v_s[n] = k_s e[n]$$

Voltage $v_c[n]$ is proportional to $v_s[n]$

$$v_c[n] = k_c v_s[n]$$

We can get the result as follow

$$v_s[n] = k_c k_s e[n]$$

System function H_{cs}

$$H_{cs} = \frac{V_c}{E} = k_c k_s$$

The complete system block diagram is as follow

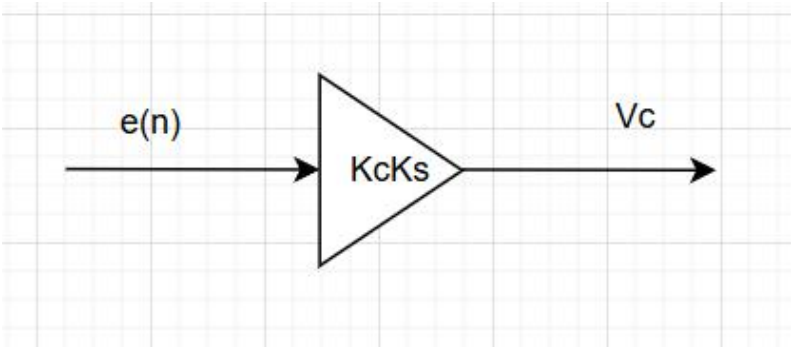

Figure 1 the diagram for the control/sensor system

Step 2

We define a function controllerAndSensorModel, which takes as input a gain kc, and outputs an instance of sf.SystemFunction which represents the controller/sensor system with this gain.

Here is the key code

```
def controllerAndSensorModel(k_c):
    return(sf.Gain(k_s*k_c))
```

Step 3

At step n, $\theta_h[n]$ is equal to $\theta_h[n-1]$ from step n-1 plus the displacement $T\omega_h[n-1]$, we get

$$\theta_h[n] = \theta_h[n-1] + T\omega_h[n-1]$$

For convenience, we use the R operator to simplify calculations, so we get

$$\Theta_h = R\Theta_h + TR\Omega_h$$

After transposition, we can get H_{int}

$$H_{int} = \frac{\Theta_h}{\Omega_h} = \frac{TR}{1-R}$$

The complete system block diagram is as follow

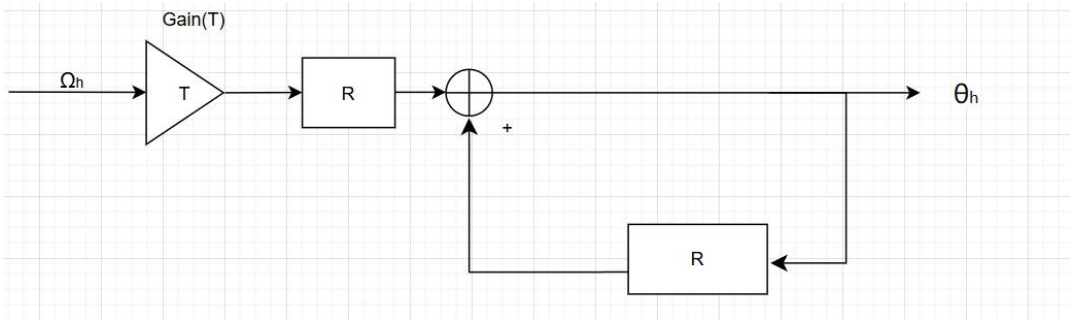

Figure 2 the system function for the integrator

Step 4

We define a method in integrator, which takes as input a timestep T (the length of time between samples) and returns an instance of sf.SystemFunction which represents an integrator with the appropriate timestep.

Here is the key code

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

Step 5

With the input $v_c[n]$,we need to get $\omega_h[n]$

Through the preconditions, we can get

$$\alpha_h[n] = k_m i_m[n]$$

$$i_m[n] = \frac{v_c[n] - v_b[n]}{v_c[n] r_m}$$

$$v_b[n] = k_b\omega_h[n]$$

At step n,$\omega_h[n]$ is equal to $\omega_h[n-1]$ from step n-1 plus the displacement $T\alpha_h[n-1]$, we get

$$\omega_h[n] = \omega_h[n-1] + T\alpha_h[n-1]$$

For convenience, we use the R operator to simplify calculations, so we get

$$\Omega_h = R\Omega_h + TRA_h$$

Substitute the equation related to A_h,we get

$$\Omega_h = R\Omega_h + TR(k_m I_m)$$

Substitute the equation related to I_m,we get

$$\Omega_h = R\Omega_h + TRk_m\left(\frac{V_c - V_b}{r_m}\right)$$

Substitute the equation related to V_b,we get

$$\Omega_h\left(1 - R + \frac{TRk_m k_b}{r_m}\right) = \frac{TRk_m}{r_m}V_c$$

Then we get H_{motor}

$$H_{motor} = \frac{\Omega_h}{V_c} = \frac{\frac{Tk_m}{r_m}R}{1 - R + \frac{Tk_m k_b}{r_m}R}$$

The complete system block diagram is as follow

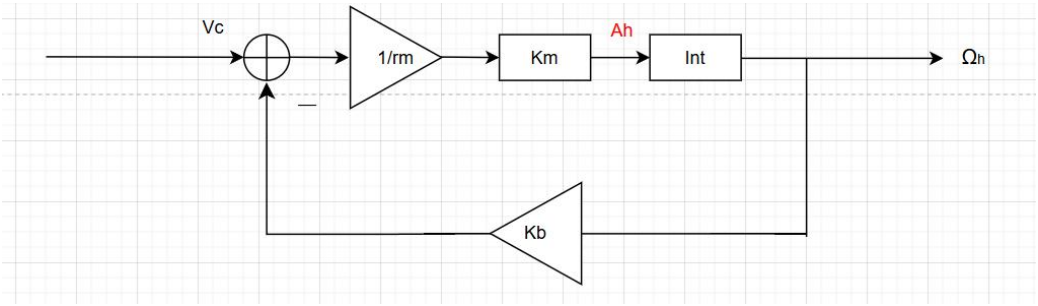

Figure 3 the system function for the motor

Step 6

We define a method motorModel, which takes the same input as integrator, but returns an instance of sf.SystemFunction which represents the motor.

Here is the key code

```
def motorModel(T):
    alpha = T * k_m / r_m
    beta = T * k_m * k_b / r_m
    H_loop = sf.FeedbackAdd(sf.R(), sf.Gain(1 - beta))
    return sf.Cascade(sf.Gain(alpha), H_loop)
```

Step7

With the input $v_c[n]$,we need to get $\theta_h[n]$

From motor, we get a system whose input is $v_c[n]$ and whose output is $\omega_h[n]$

From integrator, we get a system whose input is $\omega_h[n]$ and whose output is $\theta_h[n]$

Then all we need to do is cascade these two systems.

The complete system block diagram is as follow

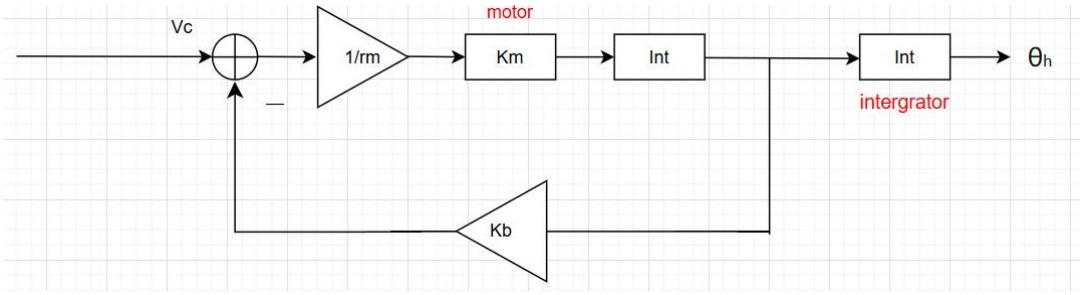

Figure 4 the system function for the plant

We can easily get

$$H_{plant} = H_{motor}H_{int} = \frac{\frac{T^2 k_m}{r_m}R^2}{(1-R)\left(1-R+\frac{Tk_m k_b}{r_m}R\right)}$$

Step 8

We define a method plantModel, which takes the same input as integrator and motorModel, and returns an instance of sf.SystemFunction which represents the entire plant.

Here is the key code

```
def plantModel(T):
    return sf.Cascade(motorModel(T),integrator(T))
```

Step 9

With the input $\theta_l[n]$,we need to get $\theta_h[n]$

From Controllers, we get a system whose input is $e[n]$ and whose output is $v_c[n]$. Among that, $e[n] = \theta_l[n] - \theta_h[n]$.

Part of $e[n]$ comes from the input $\theta_l[n]$,so we cascade the Controllers and the Plant. Then we can get a part of the output.

Another part is $\theta_h[n]$, so we should construct a degenerative feedback road.

The complete system block diagram is as follow

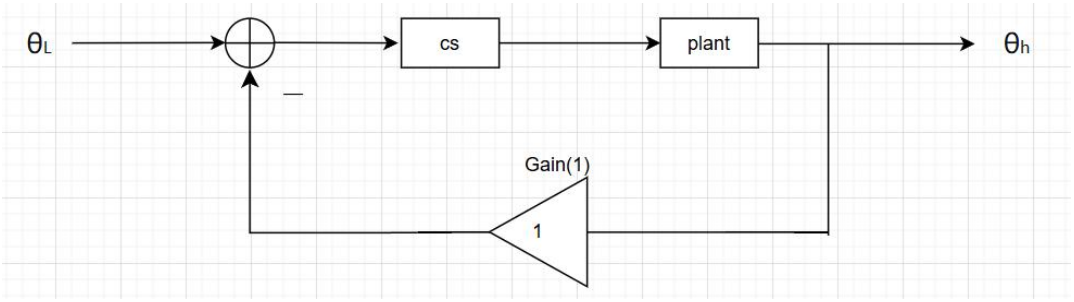

Figure 5 the system function for the composite system

The system function of forward path G

$$G = H_{cs}H_{plant} = (k_c k_s)\left(\frac{\frac{T^2 k_m}{r_m}R^2}{(1-R)\left(1-R+\frac{Tk_m k_b}{r_m}R\right)}\right)$$

The system function of the feedback path $H_f = 1$

The whole system function H

$$H = \frac{G}{1 + GH_f}$$

After simplification, the system function H

$$H = \frac{\frac{T^2 k_m k_c k_s}{r_m} R^2}{1 + \left(\frac{Tk_m k_b}{r_m} - 2\right) R + \left(1 - \frac{Tk_m k_b}{r_m} + \frac{Tk_m k_c k_s}{r_m}\right) R^2}$$

Step 10

To solve for the pole P, we set the denominator to 0.That is to say

$$1 + \left(\frac{Tk_m k_b}{r_m} - 2\right) R + \left(1 - \frac{Tk_m k_b}{r_m} + \frac{Tk_m k_c k_s}{r_m}\right) R^2 = 0$$

Using the root formula $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ and replacing $\frac{Tk_m k_b}{r_m}$, $\frac{Tk_m k_c k_s}{r_m}$, R with β, γ, $\frac{1}{z}$, we get

$$z^2 + (\beta - 2)z + (1 - \beta + \gamma) = 0$$

After solving this equation, we get

$$z = \frac{2 - \beta \pm \sqrt{\beta^2 - 4\gamma}}{2}$$

R is the final answer we need, so replace z with $\frac{1}{R}$ .Then we get

$$p = \frac{2 - \frac{Tk_m k_b}{r_m} \pm \sqrt{\left(2 - \frac{Tk_m k_b}{r_m}\right)^2 - 4\left(1 - \frac{Tk_m k_b}{r_m} + \frac{Tk_m k_c k_s}{r_m}\right)}}{2\left(1 - \frac{Tk_m k_b}{r_m} + \frac{Tk_m k_c k_s}{r_m}\right)}$$

Step 11

We define a method lightTrackerModel, which takes as input the gain to use for the controller (k_c) and the length of a timestep (T), and returns an instance of sf.SystemFunction which represents the entire light-tracking system with the specified gains and timesteps.

Here is the key code

```
def lightTrackerModel(T,k_c):
    forward_path = sf.Cascade(controllerAndSensorModel(k_c), plantModel(T))
    return sf.FeedbackSubtract(forward_path,sf.Gain(1))
```

Step 12

Find the fastest convergence and no oscillation, that is, find the k_c that minimizes the amplitude of the dominant pole.

We use the procedure optimize.optOverLine to help us search for the best k. We set numXsteps = 1000 and change the xmin, xmax

times	*xmin*	*xmax*	k_c
1	*0*	*100*	*0.6000*
2	*0*	*25*	*0.6000*
3	*0*	*1*	*0.6240*
4	*0*	*0.7*	*0.6244*
5	*0*	*0.65*	*0.6247*

Table 1 the result of procedure optimize.optOverLine

We find the k_c approaches 0.625. Then 0.625 is the best answer.

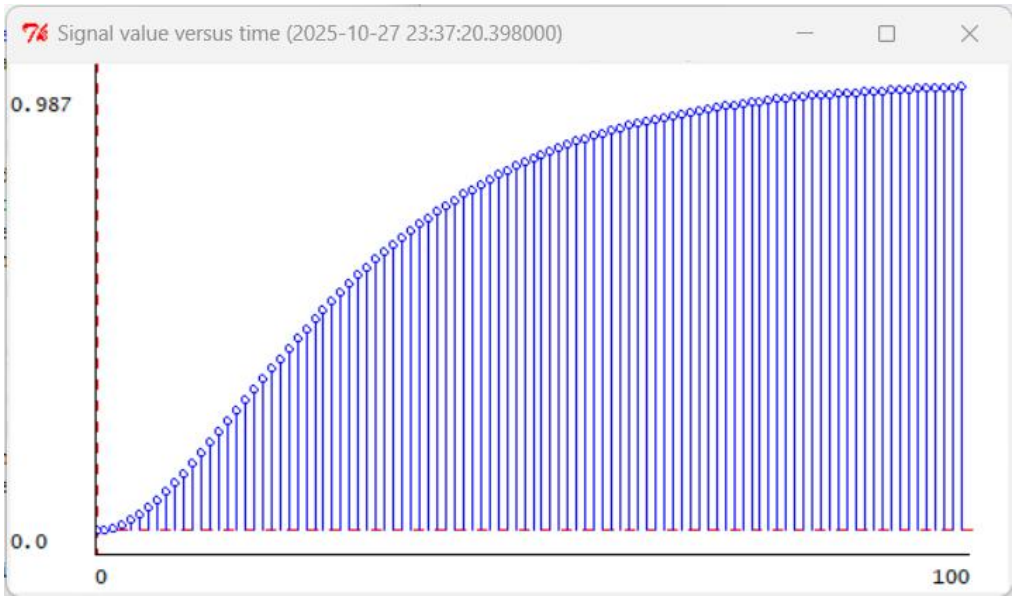

Figure 6 the situation when k=0.625

Step 13

The system is

monotonically convergent when p is a real number;

divergent when the amplitude of p exceeds 1;

oscillating convergent when p has an imaginary part but the amplitude does not exceed 1.

Judgment basis

Not oscillating means Δ greater than or equal to zero. We solved that

$$k < 0.625$$

Non-divergence means the $|p| < 1$ .We solved that

$$k < 20$$

The final result

k	*Convergence or divergence*
$0 < k < 0.625$	*monotone convergence*
$0.625 < k < 20$	*Oscillating Convergence*
$k > 20$	*divergence*

Table 2 The final result of k's range

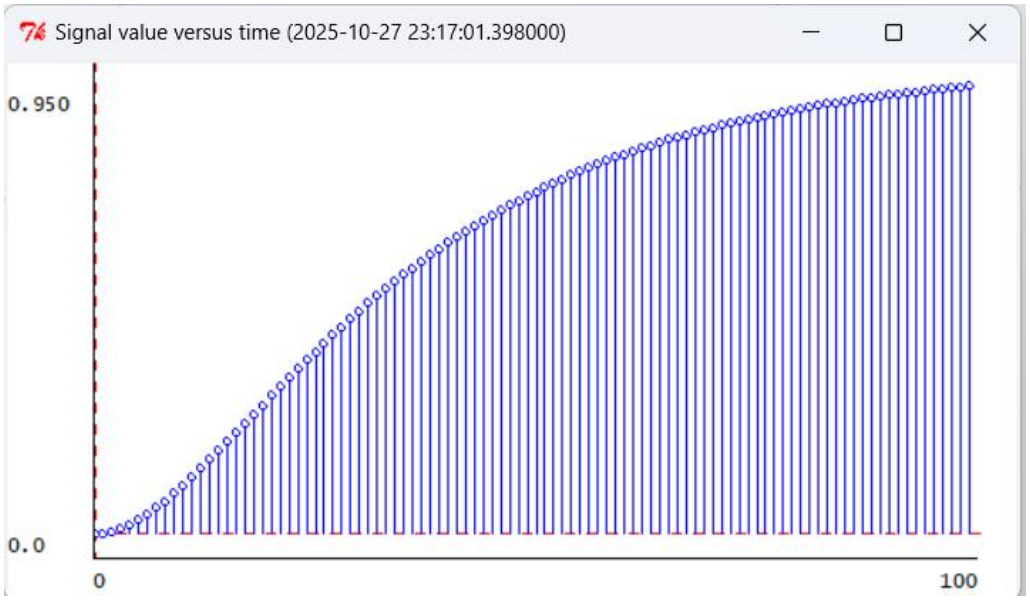

Figure 7 the situation when k_c=0.5(<0.625)

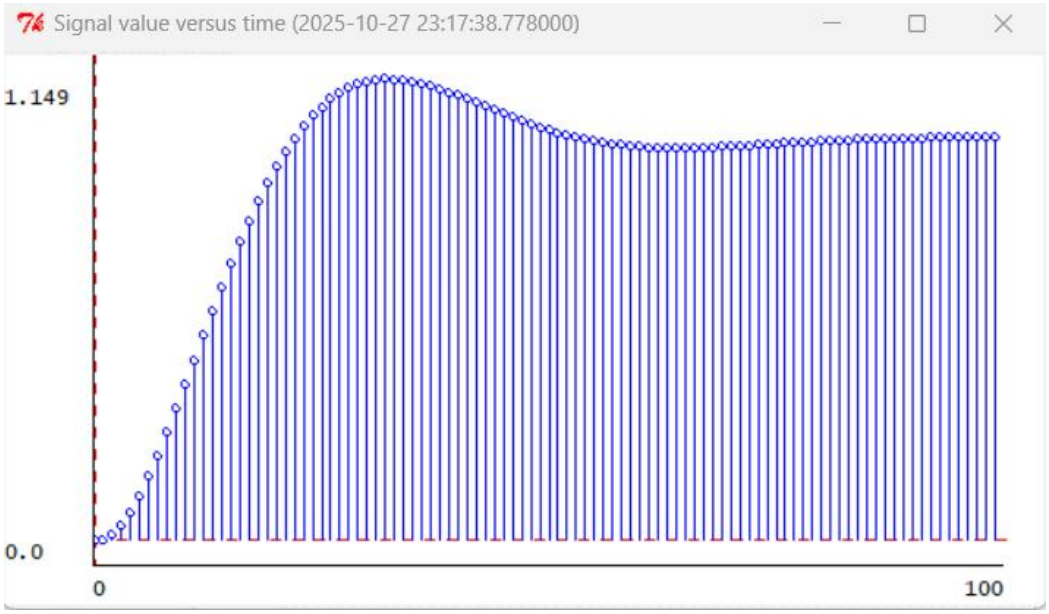

Figure 8 the situation when k_c=2(>0.625)

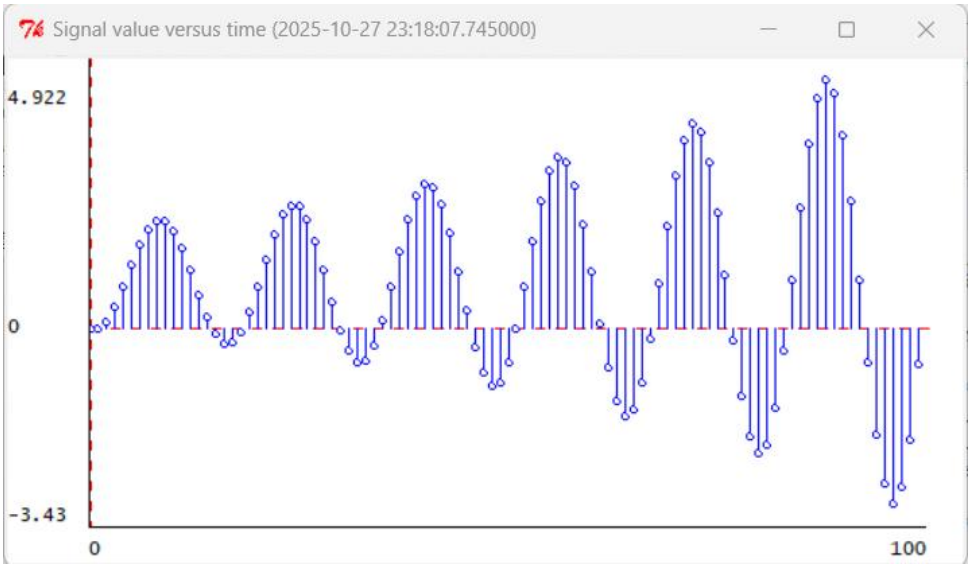

Figure 9 the situation when k_c=25(>20)

Step 14

We found that when k=0.625 is fixed, increasing T alters the convergence characteristics; for example, oscillations appear as shown in the figures below.

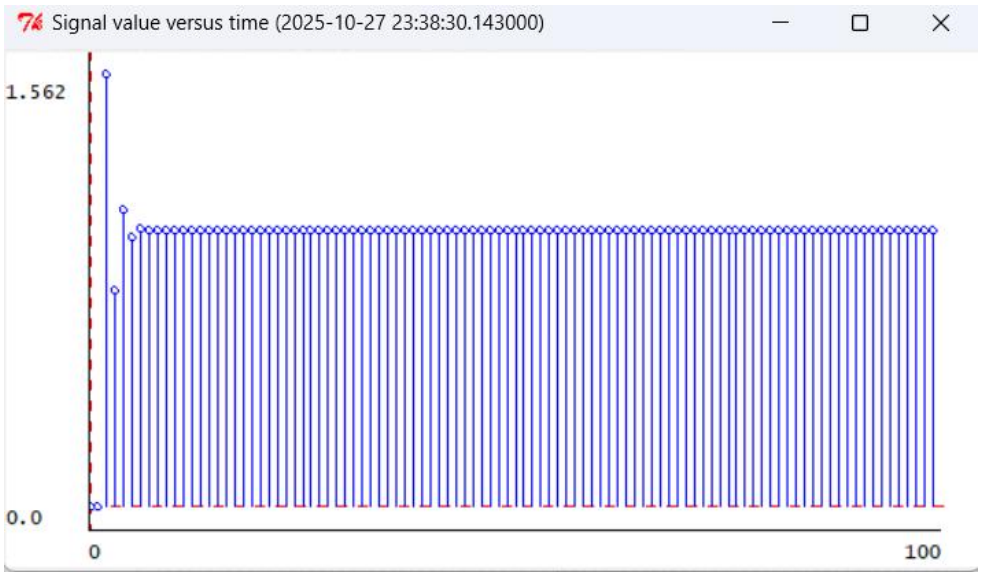

Figure 10 the situation when T=0.1

Although the number of convergence steps appears smaller, the actual time is longer because the time scale has changed. When T decreases, the poles move closer to the unit circle, resulting in a slower convergence rate.

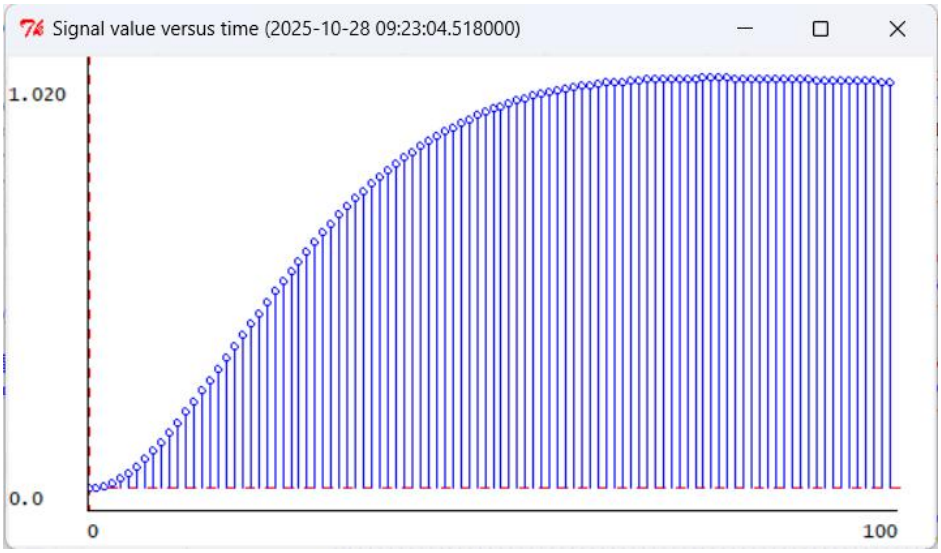

Figure 11 the situation when T=0.004

Appendix: The Description of AI Usage in the Report

During the task process, we relied on AI to help us understand the relevant concepts and problems. In the debugging of parameters, we utilized AI to efficiently achieve the goal Due to language issues, we also employed AI translation to help us quickly understand the task objectives