Step1:

Introduction

The first step is a guide to "difference equation", which requires us to build specific difference equation models for different instances. For WK.4.2.1, the difference equation we established and the correlation coefficient of the difference equation are shown in the figure below.

Problem Wk.4.2.1: Difference Equations

Determine a difference equation (with finitely many terms) for each of the systems below.

A difference equation is in the form:

$$y[n] = c_0 y[n-1] + c_1 y[n-2] + \ldots + c_{k-1} y[n-k] + d_0 x[n] + d_1 x[n-1] + \ldots + d_j x[n-j]$$

Specify the `dCoeffs`: $d_0 \ldots d_j$ and the `cCoeffs`: $c_0 \ldots c_{k-1}$ for each of the difference equations below.

Recall that we use `x` to represent the input to a system and `y` the output of the system.

Refer to Section 5.7 of the notes for examples.

For each question, enter a sequence of numbers representing the coefficients.

If one set of coefficients is empty, enter `none`, otherwise enter a sequence of numbers separated by spaces (no commas, parens, brackets, etc).

1. The output at time `n` is the sum of its inputs up to and including time `n`.
   dCoeffs (input): 1
   cCoeffs (output): 1
2. The output at time `n` is the sum of its inputs up to and including time `n-1`.
   dCoeffs (input): 1
   cCoeffs (output): 0.1
3. The output at time `n` is the sum of the scaled inputs (each input scaled by 0.1) up to and including time `n-1`.
   dCoeffs (input): 1
   cCoeffs (output): 0, 0.1

1. $y(n) = x(n) + x(n-1) + x(n-2) + \cdots$

$y(n) - y(n-1) = x(n)$

$\Rightarrow y(n) = y(n-1) + x(n)$

2. $y(n) = x(n-1) + x(n-2) + \cdots$

$y(n) - y(n-1) = x(n-1)$

$\Rightarrow y(n) = y(n-1) + x(n-1)$

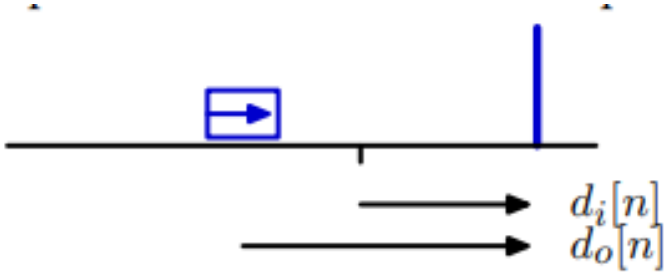
3. $y(n) = 0.1[x(n-1) + x(n-2) + \cdots]$

$y(n) - y(n-1) = 0.1\,x(n-1)$

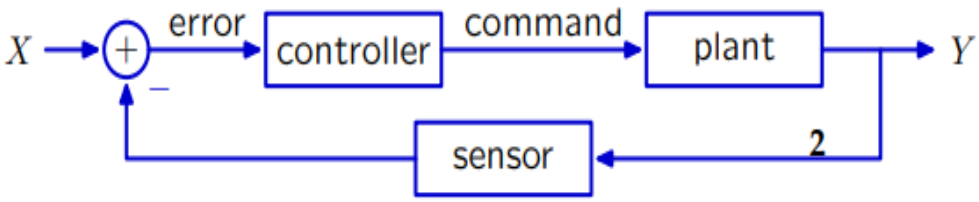
$\Rightarrow y(n) = y(n-1) + 0.1\,x(n-1)$

Step2:

Difference equations for wall finder



图一



图二

In the second step, our task is to establish the difference equation of the actual physical model. The final goal is to make the car stop at a certain distance from the wall, and the focus is on how to use the difference equation to connect multiple discrete measurement values. For the robot shown in the figure above, we first need to make clear the actual physical meaning of each discrete variable: do[n] represents the distance between the current robot and the wall, and di[n]

represents the distance between the location where we want the car to stop and the wall. (Note: di[n] is the system input and do[n] is the system output.) .V[n] is the speed of the car at time n, and we assume that the car will maintain this speed until it receives the next command at time n+1. T is the time interval over which the measurements are discretized.

Check Yourself 1.

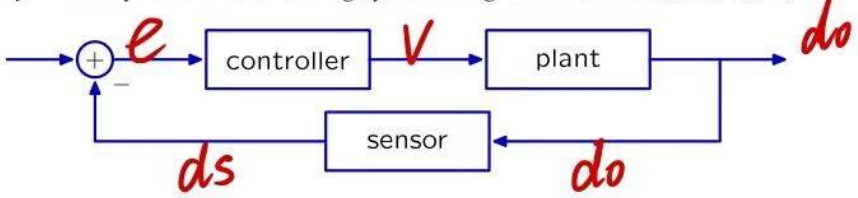
Check Yourself 1. Given the following conditions, what is the distance to the wall on step 1?

$v[0] = 1$ $\quad d_o[0] = 3$

$v[1] = 2$ $\quad d_o[1] =$ 2.9

Check Yourself 2.

Check Yourself 2. Fully label the following system diagram. Include d_o, d_i, d_s, v, e.



Determine difference equations (using constants T and k) to relate the input and output signals of the following system components.

- the controller $\quad V[n] = K e[n]$
- the model of the plant $\quad do[n] = do[n-1] - T \cdot V[n-1]$
- the model of the sensor $\quad ds[n] = do[n-1]$

Wk.4.3.1

Problem Wk.4.3.1: Wall FInder

A difference equation is in the form:

$$y[n] = c_0 y[n-1] + c_1 y[n-2] + \ldots + c_{k-1} y[n-k] + d_0 x[n] + d_1 x[n-1] + \ldots + d_j x[n-j]$$

Specify the `dCoeffs`: $d_0 \ldots d_j$ and the `cCoeffs`: $c_0 \ldots c_{k-1}$ for each of the difference equations below.

Refer to Section 5.7 of the notes for examples.

For each question, enter a sequence of numbers representing the coefficients.

If one set of coefficients is empty, enter `none`, otherwise enter a sequence of numbers separated by spaces (no commas, parens, brackets, etc).

1. Enter your answer for Check Yourself 1 2.9
2. The difference equation for the controller (so that the velocity is 5m/s when the target is 1m in front of the robot):
   dCoeffs (input): -5
   cCoeffs (output): none
3. The difference equation model of the the plant (T = 0.1 seconds).
   dCoeffs (input): 0, -0.1
   cCoeffs (output): 1
4. The difference equation model of the sensor:
   dCoeffs (input): 0, 1
   cCoeffs (output): none
5. The combined difference equation for the system (relates the output D_o to the input D_i).
   dCoeffs (input): 1, -0.5
   cCoeffs (output): 0, 0.5

According to the connection structure of the control system and the internal structure of the subsystem, if the input-output relationship is expressed as operator equation, the following results can be obtained:

$$Do = \frac{-TKR}{1 - R - TKR^2} \cdot Di$$

Convert operator equaion back to the difference equation:

Do[n]=Do[n-1]-T*K*(Do[n-1]-Di[n-1])

Checkoff1

Checkoff 1. **Wk.4.3.2**: Explain your results to a staff member.

The explanation and analysis of ***Check Yourself1***:

The physical model has been simplified appropriately in Check Yourself1. We assume that the robot changes its speed instantaneously upon receiving a command and maintains this speed until it receives the next command. Therefore, the output of the robot in the next step do 'is equal to the output of the current step do minus the rate of the robot in the current step times the interval time of the step:

do[0]-do[1]=v[0]*T, （T=0.1）

The explanation and analysis of ***Check Yourself2:***

In FIG. 2, the structure of a control system is established. Obviously, the controller and plant are cascaded structure, while the sensor is the negative feedback introduced by the system.

The subsystem is as follows：

Controller： The output of the system is a value which represent the V, the input of the system is 'error'(The net amount of input).So the Controller system is actually a multiplier that amplifies the input signal

by a factor of K to get a desired speed value V.

The difference equations of this model : V[n]=-5*e[n]

Plant: The output of the system is the distance Do from the current position of the robot to the wall, and the input of the system is the speed value V output by the controller. The relationship between Do and speed V satisfies a difference equation in *Check yourself1.*

The difference equations of this model: d0[n]=do[n-1]-T*V[n-1]

Sensor: The input of the system is the output Do of the total system, and the output of the system is actually a delay signal to Do. The sensor acts as a negative feedback network in the whole system, while the sensor system itself is just a simple delay device.

The difference equations of this model: ds[n]=do[n-1]

Wall-finder system: We connect all subsystems according to the relation shown in the structure diagram to obtain the difference equation of the wall-finder system:

do[n]=do[n-1]-T*K*(di[n-1]-di[n-2])

The explanation and analysis of ***WK.4.3.1***

To obtain the cCoeffs and dCoeffs required by each question in WK.4.3.1, we just need to transform the difference equation into a standard form and put in specific parameters.

As for the transformation of operator equation and difference equation:

We convert some variables to operators acting on inputs or outputs, We can directly change the form of the difference equation to get operator equation, or draw a block diagram of the system, and express the relationship between the output signal and the input signal through various arithmetic machines.

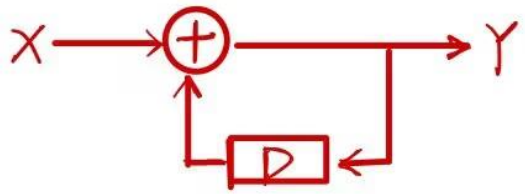
Step3:

State machines primitives and combinators

Check Yourself 3.

Check Yourself 3. Use gains, delays, and adders to draw a system diagram for the first system in tutor problem Wk.4.2.1. (That is, the tutor problem that you did before coming to lab).

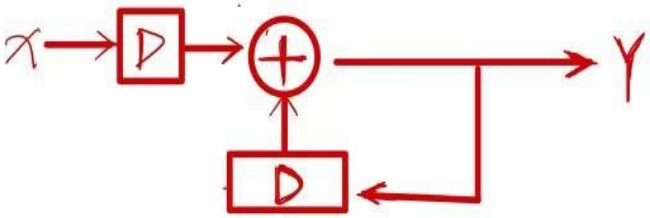
$$Y=(1+R+R^2+R^3\ldots)X \iff (1-R)Y=X$$



Check Yourself 4.

Check Yourself 4. Use gains, delays, and adders to draw a system diagram for the second system in tutor problem Wk.4.2.1.

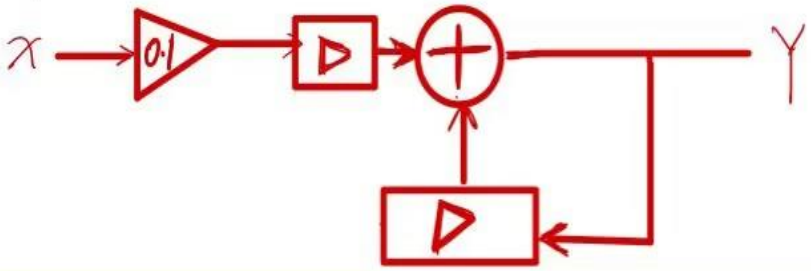
$$Y=(R+R^2+R^3\ldots)X \iff (1-R)Y=RX$$



Check Yourself 4.

Check Yourself 5. Use gains, delays, and adders to draw a system diagram for the third system in tutor problem Wk.4.2.1.

$Y = 0.1(R+R^2+R^3\cdots)X \iff 10(1-R)Y = RX$

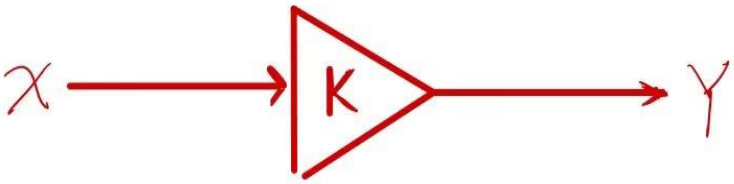


WK.4.3.3

```
import lib601.sm as sm

def accumulator(init):
    gain=sm.Gain(1)
    delay=sm.R(init)
    accum=sm.FeedbackAdd(gain,delay)
    return accum

print(accumulator(0).transduce((range(10))))

def accumulatorDelay(init):
    delay=sm.R(init)
    accum=sm.Cascade(delay,accumulator(init))
    return accum

print(accumulatorDelay(0).transduce((range(10))))

def accumulatorDelayScaled(s,init):
    accum=sm.Cascade(sm.Gain(s),accumulatorDelay(init))
    return accum

print(accumulatorDelayScaled(0.1,0).transduce((range(10))))
```

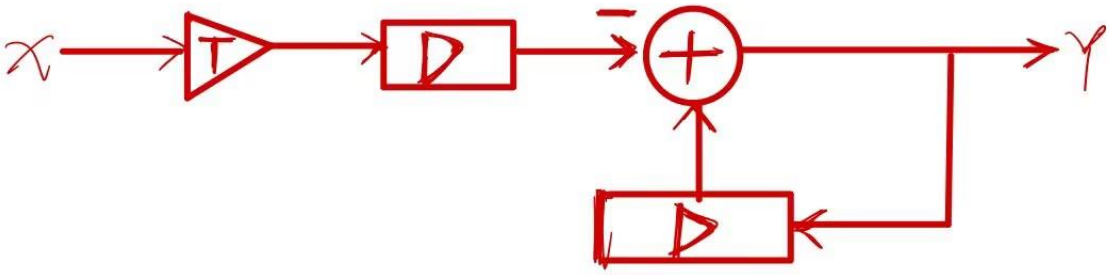
Check Yourself 6.

Check Yourself 6. Use gains, delays, and adders to draw a system diagram for the **controller** in the wall-finder system.



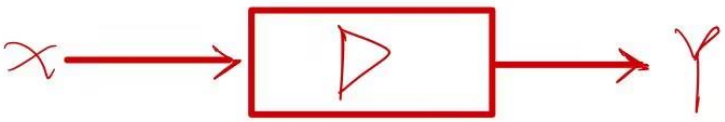
Check Yourself 7.

Check Yourself 7. Use gains, delays, and adders to draw a system diagram for the **plant** in the wall-finder system.



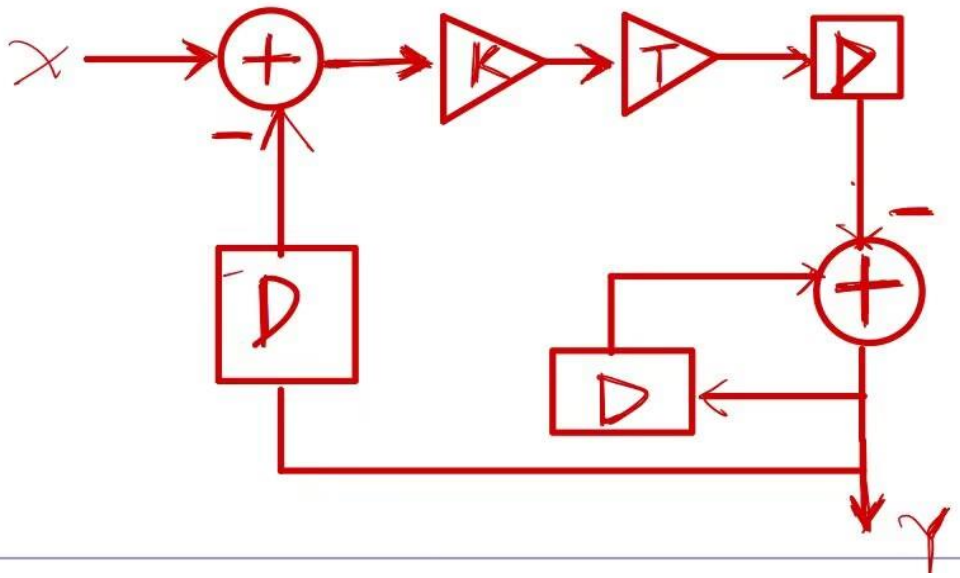
Check Yourself 8.

Check Yourself 8. Use gains, delays, and adders to draw a system diagram for the **sensor** in the wall-finder system.



Check Yourself 9.

Check Yourself 9. Connect the previous three component systems to make a diagram of the wall-finder system. Label all the wires. Draw boxes around the controller, plant, and sensor components.



WK.4.3.5

```
def plant(T, initD):
    sm1=sm.Cascade(sm.Gain(-T),sm.R(0))
    return sm.Cascade(sm1,sm.FeedbackAdd(sm.Gain(1),sm.R(initD)))

def controller(k):
    gain=sm.Gain(k)
    return gain

def sensor(initD):
    delay=sm.R(initD)
    return delay

def wallFinderSystem(T, initD, k):
    sm1=sm.Cascade(controller(k),plant(T,initD))
    sm2=sensor(initD)
    return sm.FeedbackSubtract(sm1,sm2)
```

Check Yourself 10.

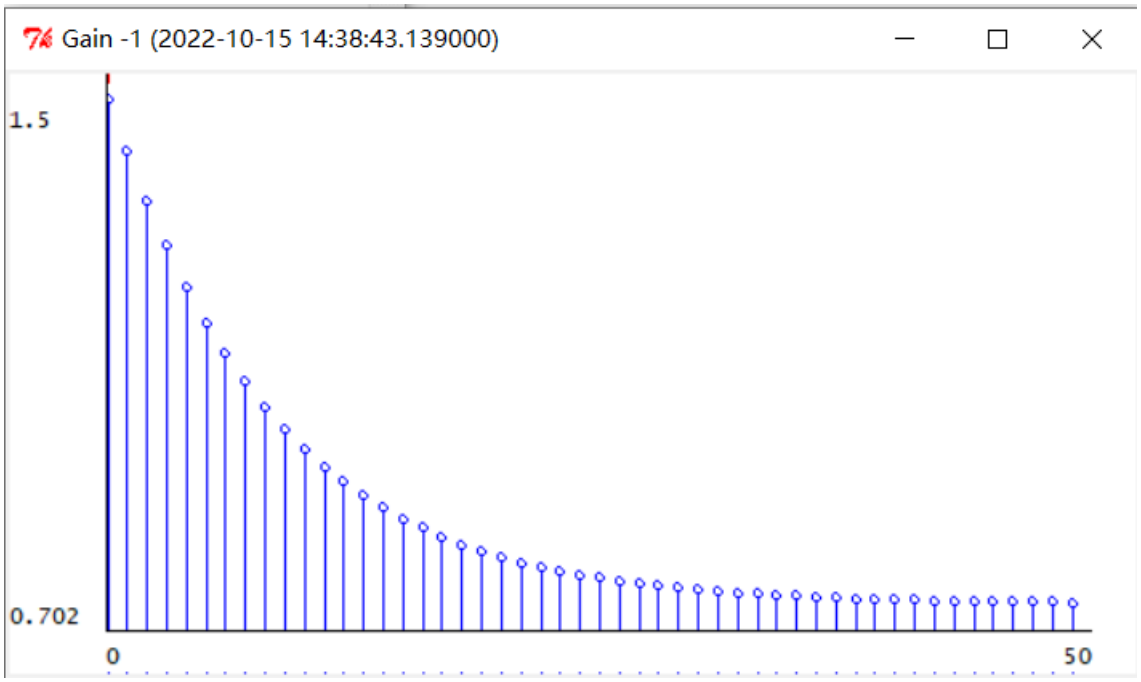


图 1（k=1）

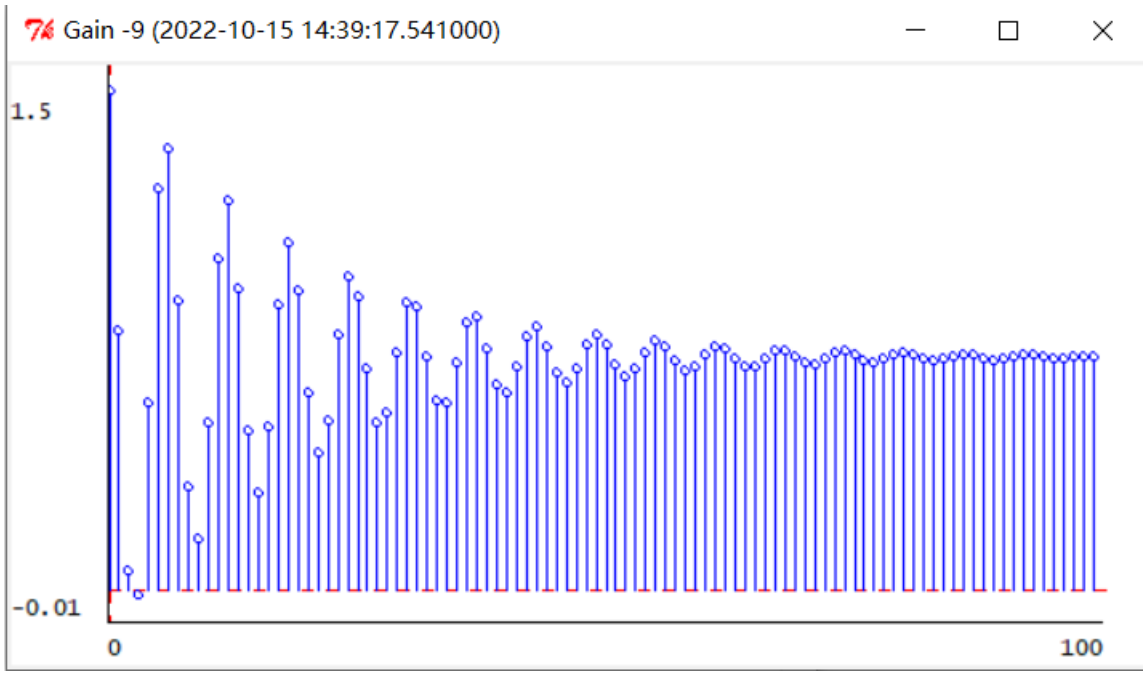


图 2（k=9）

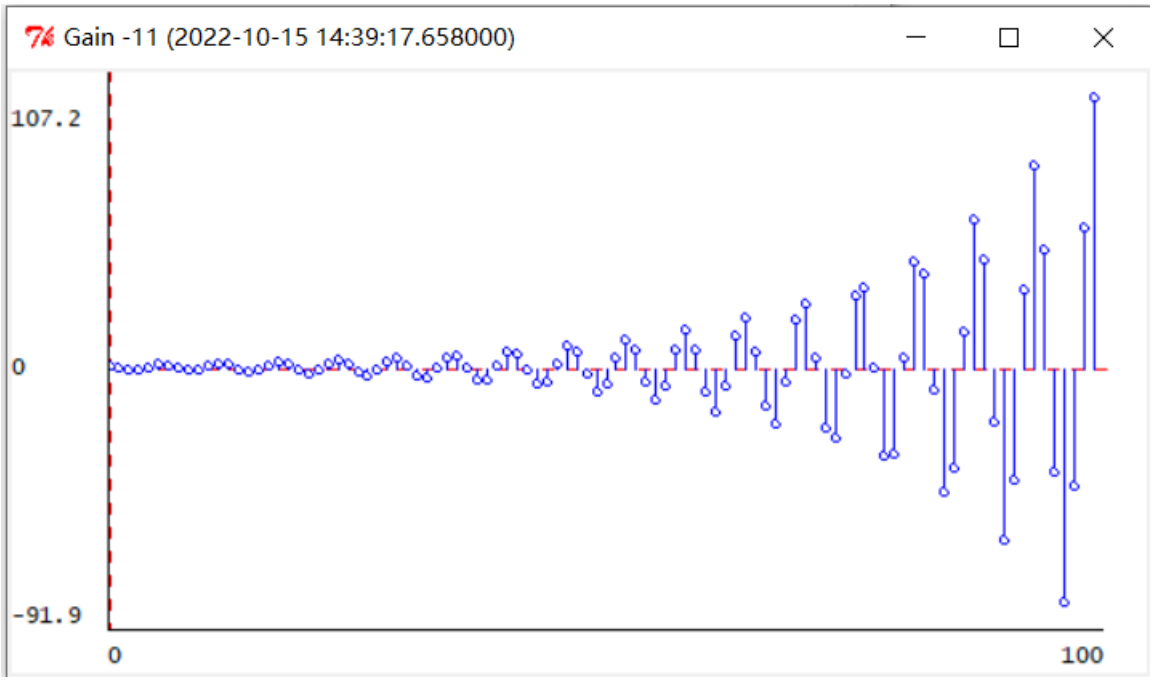


图 3（k=11）

WK.4.3.6

Problem Wk.4.3.6: Wall Finder Gains

Enter gains that produce the following types of behaviors for the wall-finder system (with time step T of 0.1).

1. The distance converges monotonically (without oscillation): 1
2. The distance oscillates and converges (approaches a finite value): 9
3. The distance oscillates and diverges (grows without bound): 11

Checkoff2

The explanation and analysis of Check Yourself3:

The block diagram of this operator equation contains an adder and a Feedback network， The feedback network is a unit delayer, so as to achieve the accumulator effect.

The explanation and analysis of Check Yourself4:

The block diagram of this model can be directly improved on the basis of the block diagram of *Check Yourself3*. As accumulator needs to accumulate to X[n-1] instead of X[n], a unit delay accumulator is added at the input end to delay the input X by one unit before input to the whole system.

The explanation and analysis of Check Yourself5:

The block diagram of this model is also changed on the block diagram in check yourself4, which requires adding a multiplier in the input segment to achieve the proportional scaling of the input signal.

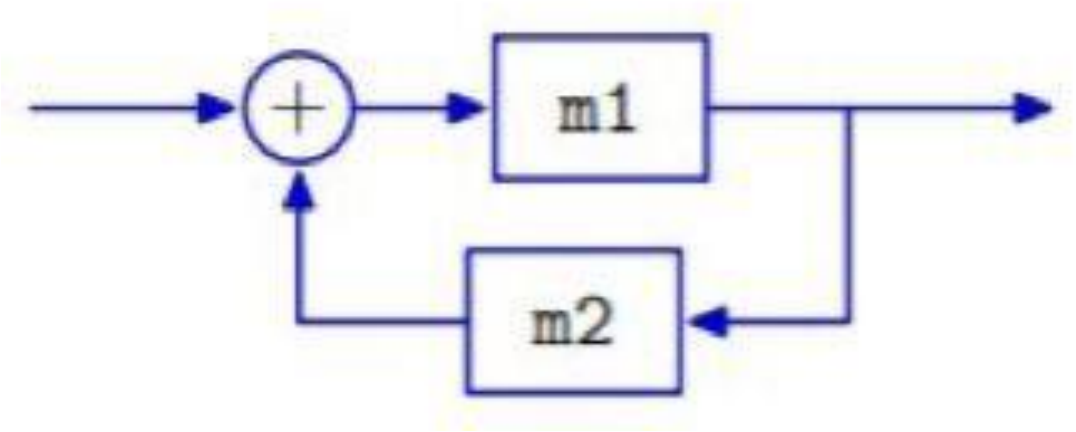
The explanation and analysis of WK.4.3.3:

In WK.4.3.3, python is used to establish the above three operator equation functions. We need to use the sm module in lib601 library for the relationship between the computers.

The following figure shows a block diagram of *FeedbackAdd*

In the figure, m1 and m2 represent state machine instances. The first parameter of sm.FeedbackAdd corresponds to m1 and the second

parameter corresponds to m2



accumulator: The return value of the function is an instance of sm.FeedbackAdd.m1 should be an instance of sm.Gain in this function. sm.Gain functions as a multiplier, and the output is K times the input. In this function, let K=1, indicating that the output is equal to the input. m2 is an instance of sm.R, which plays the role of unit delay.

accumulatorDelay: The return value is an sm.Cascade instance. The first argument of the sm.Cascade instance is the unit delay (an instance of sm.R), and the second argument is the *accumulator* which is a function we just defined.

accumulatorDelayScaled: The return value of the function is also a sm.Cascade instance. By using the function defined above, directly cascade a multiplier (the instance of sm.Gain) with accumulatorDelay function to scale the input by k times.

The explanation and analysis of Check Yourself6, Check Youself7 and Check Yourself8:

In step2, we have established the difference equation model of controllor, plant and sensor subsystems, and directly transformed the difference equation into operator equation:

controllor: Y=KX

plant: Y=RY-TRX → Y=$\frac{TR}{R-1}X$

sensor:Y=RX

Wall-finder system: $Y=\frac{-TKR}{1-R-TKR^2}\cdot X$

The explanation and analysis of ***WK.4.3.5***

(Build wall-finder system in python):

According to the block diagram and operator equation:

Controller:Returns an instance of sm. gain. The parameter K is the scale of the input

Plant: Returns an sm.Cascade instance. The first argument is also a Cascade instance to cascade a multiplier and unit delay, and the second argument is a feedback adder.

Sensor: Returns an instance of sm.R that represents the role of the unit delayer

Wall-finder system: As a whole, the system is a feedback adder. For the first parameter m1 state machine of the sm.FeedbackAdd instance, it is the cascade of the controller return value and the plant return

value, and the second parameter m2 state machine is the sensor return value.

The explanation and analysis of ***Check Yourself10***:

QUESTION: The relationship between k value and the final distance of the robot from the wall.

We first analyze qualitatively, and the k value first affects the speed of each step robot. When k is small, the robot will eventually stabilize at d0 away from the wall, and the speed of the robot decreases gradually in the process of movement. Therefore, the image of do-n (n is the sampling point) should be a single subtraction concave function, which tends to be flat. When k is relatively large, we can imagine that the robot will definitely jump di in a certain step. At this time, according to the difference equation, the speed will be reversed. If the distance of the robot from the wall is greater than di in the next step, the robot will move forward again, and so on.

It's an oscillatory process, but does the robot end up at exactly di? It depends on the size of k.

do-n images are observed in the program by changing the value of k each time. We found three k values, k=1, k=9 and k=11, which correspond to three cases of monotone convergence, oscillation convergence and oscillation divergence respectively.

Step4

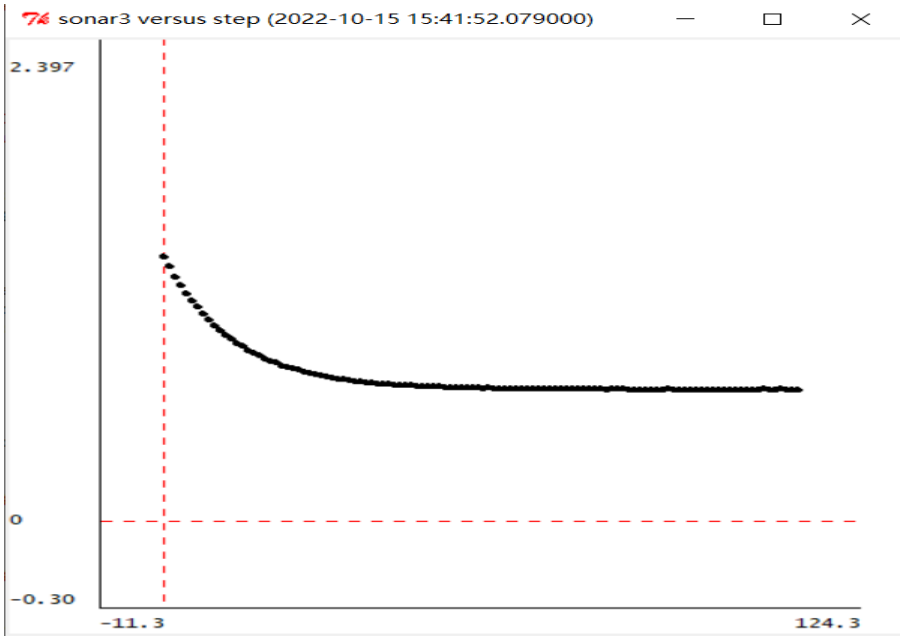
On the simulated robot

The task in Step4 turns to control the actual tracking movement of the car, and debugging through simulation

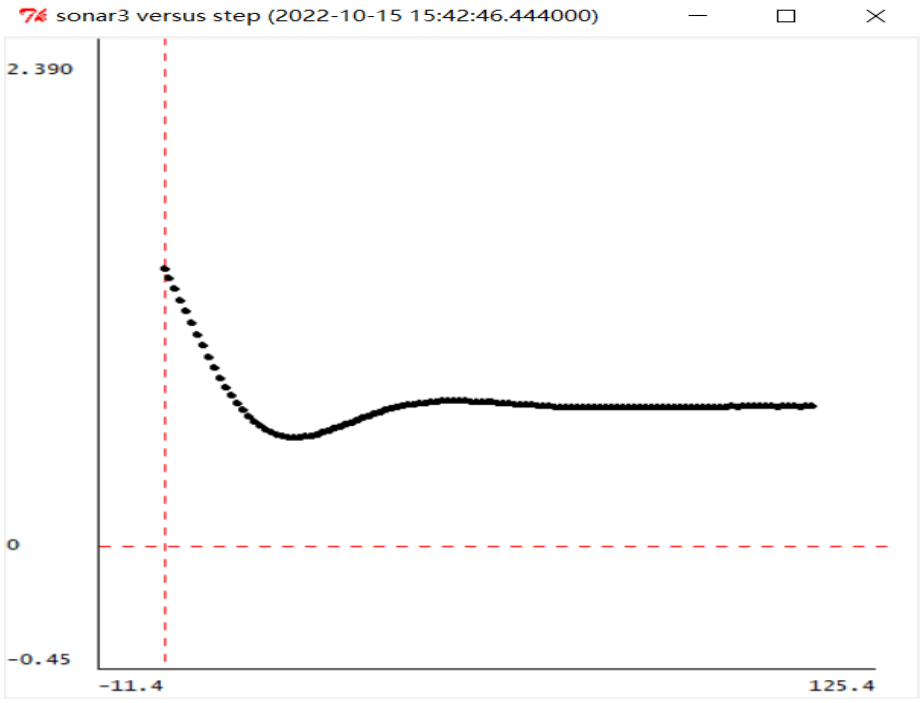
Class definition in python:

```
# Input is output of Sensor machine (below); output is an action.
# Note that this machine must also compute E, the error, and output
# the velocity, based on that.
class Controller(sm.SM):
    def getNextValues(self, state, inp):
        return (state,io.Action(fvel=inp-dDesired,rvel=0))

# Input is SensorInput instance; output is a delayed front sonar reading
class Sensor(sm.SM):
    def __init__(self, initDist, numDelays):
        self.startState = [initDist]*numDelays
    def getNextValues(self, state, inp):
        print inp.sonars[3]
        output = state[-1]
        state = [inp.sonars[3]] + state[:-1]
        return (state, output)

mySM = sm.Cascade(Sensor(1.5, 1), Controller())
mySM.name = 'brainSM'

################################################################################
```

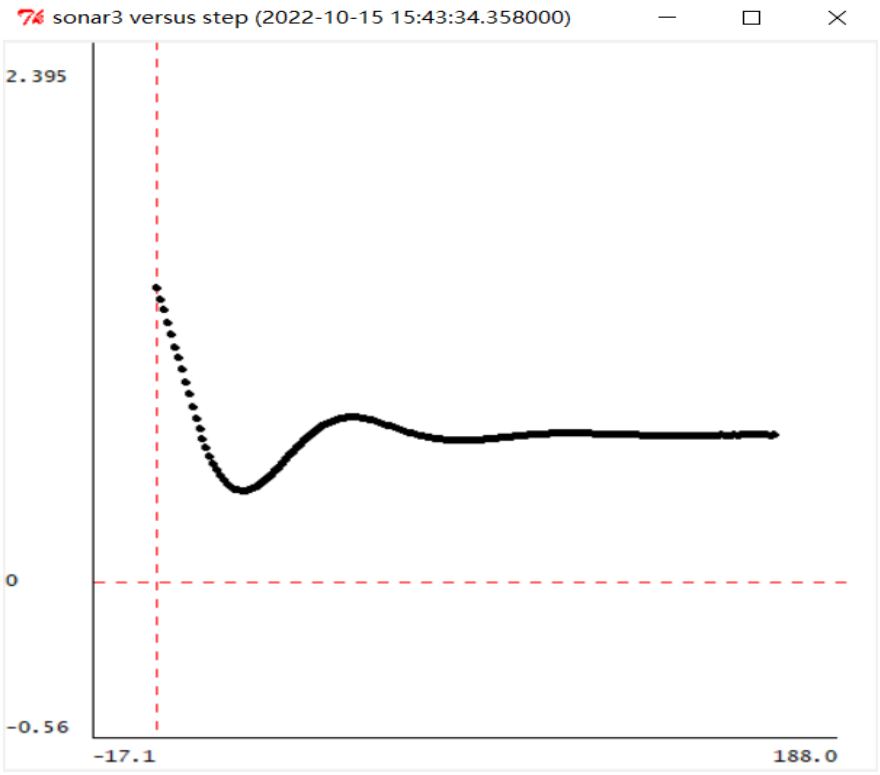
Check Yourself 11.



(k=-1)



(k=-9)



(k=-11)

Checkoff3

Compare the image in Check yourself10 and Check yourself11:

When k=1, both images converge monotonically,

When k=9, although both images are oscillatory convergence, it is obvious that in check yourself10, the amplitude of image oscillation is larger and the frequency of oscillation is higher.

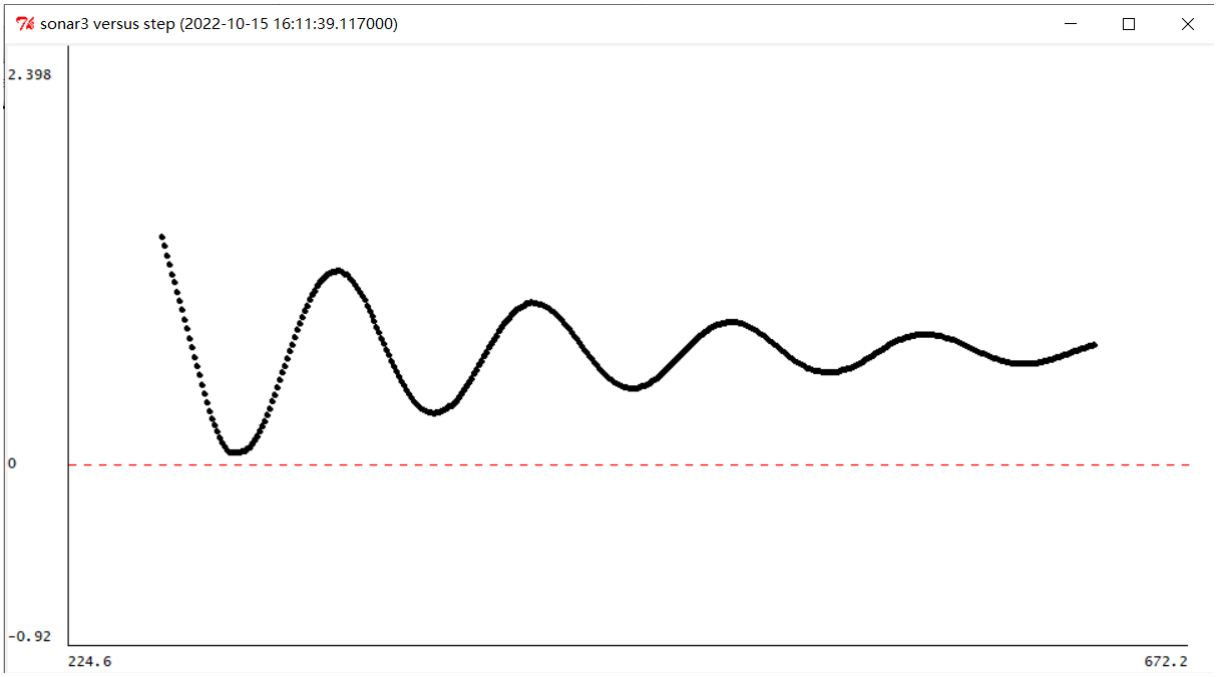
When k=11, The image in check yourself10 is oscillating and diverging while the image in check yourself11 is oscillating and converging.

Analysis:

From the perspective of program parameter adjustment:

(Increasing the value of k in units of 1)

By comparing the difference of the images, it can be concluded that the k range in check yourself11 is greater than that in check yourself10 when the image converges.



(k =-18)

When k=18, the image still oscillates and converges. Continue to increase the value of k, the car will hit the wall in the simulation mapIf the car hits the wall on the simulation map, the image cannot be drawn effectively, but it can be predicted that the K value of image divergence will be much larger than that in check yourself10.)

From the theoretical level:

Let's go back to our original difference equation model:

do[n]=do[n-1]-T*V[n-1]

The operator equation: $Do=\frac{-TKR}{1-R-TKR^2}\cdot Di$

After replacing R by 1/z, we can find the poles analytically

:$z^2-z-TK=0$

The roots of this polynomial: $\frac{1}{2}\pm\frac{1}{2}\sqrt{1+4KT}$

When the system converges, the absolute value of the extreme point is less than 1, modulo less than 1 for the complex number.

Therefore,In check yourself10 (T=0.1) :

when k>0, the system diverges monotonically;

when -2.5 >k>-10 , The system oscillates but converges.

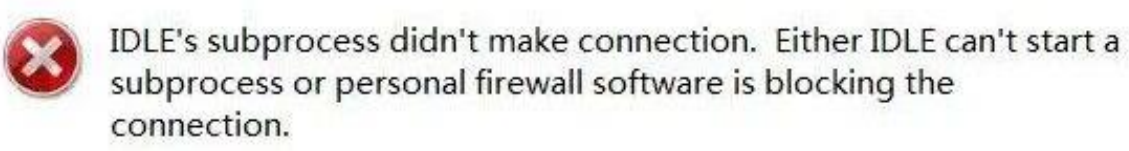
When k < -10, The system oscillates and diverges

But in check yourself11: The interval T between adjacent detection values of the car's sensor is less than 0.1, so the k value of the critical point of convergence and divergence of the system is smaller than -10.

Therefore, when k=-11, the image in *check yourself10* diverges while the image in *check yourself11* does not diverge.

Summary

1. Through the study of this experiment, I have experienced how to establish a simple difference equation model and operator equation of a discrete system. When faced with a more complex discrete system, a very important point is to use the PCAP thought, will be a discrete system as the combination of several simple subsystem, make sure the function of each subsystem in complex system, establish a subsystem model of internal structure, between different subsystems to build the connection between the input and output, Find the connections between subsystems, such as cascade, parallel, feedback, etc., and finally integrate subsystems into complex systems. At the same time, the idea of PCAP is also reflected in how to abstract a real physical model, and through appropriate simplification, the physical quantity is abstracted into measurable and computable discrete variables.
2. In the experiment of step3, check yourself10 requires us to get the image of do-n under three k values, but the following error occurs when the plotD function in designlab04work.py is called:



We checked the code several times and found no grammatical or logical problems. We also considered whether the library file was missing, so we re-installed the lib601 library file, and found that the problem was still not solved.Finally, we searched the "Subprocess Startup Error" in the error prompt online and found a solution:

解决方案:

1>修改pyshell.py文件

python idle报错修复方案1

修改python目录/Lib/ idlelib\PyShell.py文件，将1379行的use_subprocess变量值修改为False，即启动idle不使用子进程。这样子就可以避免idle启动的时候启动子进程失败进而崩溃。

注意：这会导致在idle中无法启用子进程，如果你不需要进行多进程编程，可以忽略这个缺陷。

2>修改与内建文件重名的自定义文件

如果你自己写了一个copy.py文件，并且保存在了python的库文件夹下，并且好死不死还运行过导致自动生成了.pyc文件。那么idle在启动的时候，就会调用你写的那个copy.py文件，导致崩溃。

如果是这样的情况，就要查一下是否有这样的和python内建库重名的文件了，删掉就OK了。

但是python有那么多的内建库，找起来还是，emmm比较麻烦的。就当是为自己不好的命名习惯买单了。

3>防火墙放行

如果前两个方法都试过了，无效的话可以看看这个方法。

打开控制面板>系统与安全>Windows防火墙>允许的程序，看看列表里面有没有python，如果没有的话，使用下面的"允许运行另一程序"按钮，在python的安装目录下，添加pythonw.exe和python.exe。最后在python对应的项目后面加上勾。

3. In step4, it is required to analyze the reasons for the different convergence of the drawn images with the same k value in check yourself10 and check yourself11, which is a difficult point in the experiment. This needs to start from the principle, back to the theoretical level of thinking . $Y=\frac{-TKR}{1-R-TKR^2}\cdot X$ Represents the relationship between the input quantity and the output quantity in the

wall-finder system.After consulting relevant theoretical data, we learned that $\frac{-TKR}{1-R-TKR^2}$ Representative system function，For a system function, we can find the zeros and poles of the system function. The method of finding the pole is also a difficult one. The method is to find the zero of the z polynomial by setting 1/z=R. The convergence and divergence of system functions depend on the size of poles, so we find the range of k under critical conditions from the principle formula, which is a physical quantity related to T. Therefore, it is the difference in T between adjacent steps of the system that leads to the difference in convergence for the same value of k.

实验报告要求：

1、 上面为实验报告的电子模板；

2、 根据实验讲义，记录各个步骤（Step）需要记录的关键内容、实验结果、仿真结果（比如仿真截图，测试代码 IDLE 运行结果等）、实验结论、遇到的问题及解决方案等，简单记录即可，按照讲义的实验思路一步步完成，可以类似于实验 LOG 的书写。课下准备实验以及实验室做实验过程中可以边做边写，没有什么需要记录的 Step 可以略过；

3、 实验讲义中的 Check Yourself 和 Checkoff 需要在报告中完成；

4、 系统框图、数据表格等请记录清晰，公式使用公式编辑器输入，需要展示的程序片段可以复制粘贴或者截图；

5、 最后总结部分（Summary）写一写实验总结和心得，遇到的问题和解决方案等；

6、 实验报告内容推荐使用全英文书写，不好记录或者表述的中英文结合或者中文也可以。