

6.01 Midterm 1 Makeup : Fall 2010

Name: Michael Plamondon

Section: Morning

Enter all answers in the boxes provided.

During the exam you may:

- read any paper that you want to
- use a calculator

You may not

- use a computer, phone or music player

For staff use:

1.	11	/12
2.	14	/16
3.	16	/16
4.	15	/21
5.	13	/20
6.	12	/15
total:	81	/100

1 OOP (12 points)

The following definitions have been entered into a Python shell:

```
class Account:
    chargeRate = 0.01
    def __init__(self, start):
        self.value = start
    def debit(self, amount):
        debitAmt = min(amount, self.value)
        self.value = self.value - debitAmt
        return debitAmt
    def deposit(self, amount):
        self.value += amount
    def fee(self, baseAmt):
        self.debit(baseAmt * self.chargeRate)
    def withdraw(self, amount):
        if self.value >= 10000.0:
            self.fee(amount/2.0)
        else:
            self.fee(amount)
        return self.debit(amount)

class Checking(Account):
    chargeRate = 0.05
    def deposit(self, amount):
        if self.value <= 1:
            Account.deposit(self, (1-self.chargeRate) * amount)
        else:
            Account.deposit(self, amount)
```

Assume that the following expressions have been evaluated:

```
Eric = Checking(3000.0)
Ellen = Account(3000.0)
```

2000 * .05
\$100 fee

11/12

Write the values of the following expressions. Write None when there is no value; write Error when an error results and explain briefly why it's an error. Assume that these expressions are evaluated one after another (all of the left column first, then right column).

Eric.withdraw(2000.0)

\$100 fee
2000

Eric.value

900

Eric.withdraw(1000.0)

Fee charged on full amt \$45
855 -1

Eric.value

0

Eric.deposit(4000.0)

\$200 fee here too
None

Eric.value

3800
nothing printed

Ellen.withdraw(2000.0)

2000 *fee = \$20*

Ellen.value

980

Ellen.withdraw(1000.0)

970 *fee \$10*

Ellen.value

0

Ellen.deposit(4000.0)

None *no fee here*

Ellen.value

4000
still nothing printed

2 So who kicks b*** (16 points)

Here are some class definitions, meant to represent a league of football teams.

```
class Team:
    def __init__(self, name, wins, losses, pointsFor, pointsAgainst):
        self.name = name
        self.wins = wins
        self.losses = losses
        self.pointsFor = pointsFor
        self.pointsAgainst = pointsAgainst

class League:
    def __init__(self):
        self.teams = {}
    def addTeam(self, team):
        self.teams[team.name] = team
    def updateGame(self, teamName, ptsFor, ptsAgin):
        # to be filled in
    def computeStat(self, proc, filt):
        return [proc(team) for team in self.teams.values() if filt(team)]
```

Imagine instantiating these classes by:

```
Pats = Team('Pats', 5, 1, 150, 100)
Ravens = Team('Ravens', 4, 2, 80, 30)
Colts = Team('Colts', 1, 4, 100, 200)
```

```
NFL = League()
NFL.addTeam(Pats)
NFL.addTeam(Ravens)
NFL.addTeam(Colts)
```

We would like to be able to update our information by adding in new game data. For example, if the Pats beat the Colts, by a score of 30 to 15, the record for the Pats should include another win, and an updated record of points for and points against; similarly for the Colts. We would do this by calling

```
NFL.updateGame('Pats', 30, 15)
NFL.updateGame('Colts', 15, 30)
```

Write a Python procedure that will complete the definition of `updateGame`. You may assume that all teams exist in the instance of the league, and that there are no ties, only wins and losses. Please make sure that the indentation of your written solution is clear.

```
def updateGame(self, teamName, ptsFor, ptsAgin):  
    if ptsFor > ptsAgin: #win  
        self.teams[teamName].wins += 1  
        self.teams[teamName].pointsFor += ptsFor  
    else: #loss; can be no ties  
        self.teams[teamName].losses += 1  
        self.teams[teamName].pointsAgainst += ptsAgin
```

U = tab

Assume that the NFL has been defined as above, but with more teams. Write an expression using `computeStat`, to be evaluated by the Python interpreter, that will return a list of each team's name and its wins, for all teams in the NFL. (It is okay to define and use helper functions.) For the example instance defined above, your expression should return the list:

[['Pats', 5], ['Ravens', 4], ['Colts', 1]]

not really sure
how this
is used

~~NFL.computeStat(lambda x: x.wins, 1==1)~~
 NFL.computeStat(lambda x: [x.name, x.wins], 1==1)
 procedure
 - 1

Write an expression using `computeStat`, to be evaluated by the Python interpreter, that will return a list of the `pointsFor` for the Pats and the Ravens. (It is okay to define and use helper functions.) For the example instance defined above, your expression should return the list:

[150, 80]

NFL.computeStat(lambda x: [x.pointsFor], team == 'Pats' or 'Ravens')
 procedure
 - 1

- does it use lambda
 - yes - proc

3 Will it or won't it? (16 Points)

16

For each difference equation below, say whether, for a unit sample input signal:

- the output of the system it describes will diverge or not as n approaches infinity, assuming that $x[n], y[n] = 0$ for $n < 0$, *bounded*
- the output of the system it describes (a) will always be positive, (b) will alternate between positive and negative, or (c) will have a different pattern of oscillation

Part 1:

$$3y[n] + 4y[n-1] = x[n-3]$$

diverge? Yes or No

Yes

Why?

The magnitude of the root is > 1 so the output will be unbounded

positive/alternate/oscillate

alternate

Why?

The pole is ^{real} negative, so the output signal will, after finitely many steps begin to alternate sign

Part 2:

$$y[n] = y[n-1] - \frac{1}{4}y[n-2] + x[n-1]$$

diverge? Yes or No

No

Why?

The magnitude of the root is < 1 so it will be bounded, thus not diverge

positive/alternate/oscillate

Positive

Why?

The root is real and positive so the output signal will, after finitely many steps begin to increase or decrease monotonically.

→
See back

$$3y[n] + 4y[n-1] = x[n-3]$$

$$3Y + 4YR = XR^3$$

$$Y(3+4R) = XR^3$$

$$\frac{Y}{X} = \frac{R^3}{3+4R}$$

$$^n[4,3]$$

$$\frac{1}{z^2} \text{ reverse } [3,4]$$

$$3z + 4 = 0$$

$$3z = -4$$

$$z = -\frac{4}{3}$$

real \ominus - will begin to alternate sign
magnitude > 1 output may be unbounded

$$y[n] = y[n-1] - \frac{1}{4}y[n-2] + x[n-1]$$

$$Y = YR - \frac{1}{4}YR^2 + XR$$

$$Y(1 - R + \frac{1}{4}R^2) = XR$$

$$\frac{Y}{R} = \frac{R}{\frac{1}{4}R^2 - R + 1}$$

$$R = \frac{1}{z}$$

$$= \frac{1}{z}$$

$$\frac{\frac{1}{4}\left(\frac{1}{z}\right)^2 - \frac{1}{z} + 1}{}$$

$$z^2$$

$$z^2$$

$$= \frac{z}{\frac{1}{4} - \frac{z}{z} + \frac{z^2}{z}}$$

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$2a$$

$$\frac{1 \pm \sqrt{1 - 4 \cdot 1 \cdot \frac{1}{4}}}{2}$$

$$2$$

$$\sqrt{0} = 0$$

$$\frac{1}{2}$$

real \oplus increase/decrease monotonically
magnitude < 1 will be bounded

4 Grow, baby, grow (21 Points)

You and your colleague in the Biology Department are growing cells. In each time period, every cell in the bioreactor divides to yield itself and one new daughter cell. However, due to aging, one quarter of the cells die after reproducing (don't worry about the details of how accurately this models real cell division).

We can describe this system with the following difference equation. We let P_o denote the number of cells at each time step.

Then

$$P_o[n] = 2P_o[n-1] - 0.25P_o[n-2]$$

Suppose that $P_o[0] = 10$ and $P_o[n] = 0$ if $n < 0$. What are the first few values for the number of cells (note that while not physically realistic, our model might provide fractional answers)?

$$P_o[0] = 10$$

$$P_o[1] = 20$$

$$P_o[2] = 37\frac{1}{2}$$

$$P_o[3] = 70$$

Your goal is to create a constant population of cells, that is, to keep P_o constant at some desired level P_d . You are to design a proportional controller that can add or remove cells as a function of the difference between the actual and desired number of cells. Assume that any additions or deletions at time n are based on the measured number of cells at time $n-1$. Denote the number of cells added or removed at each step P_{inp} . Derive the difference equations that govern this closed loop system. *what we just did*

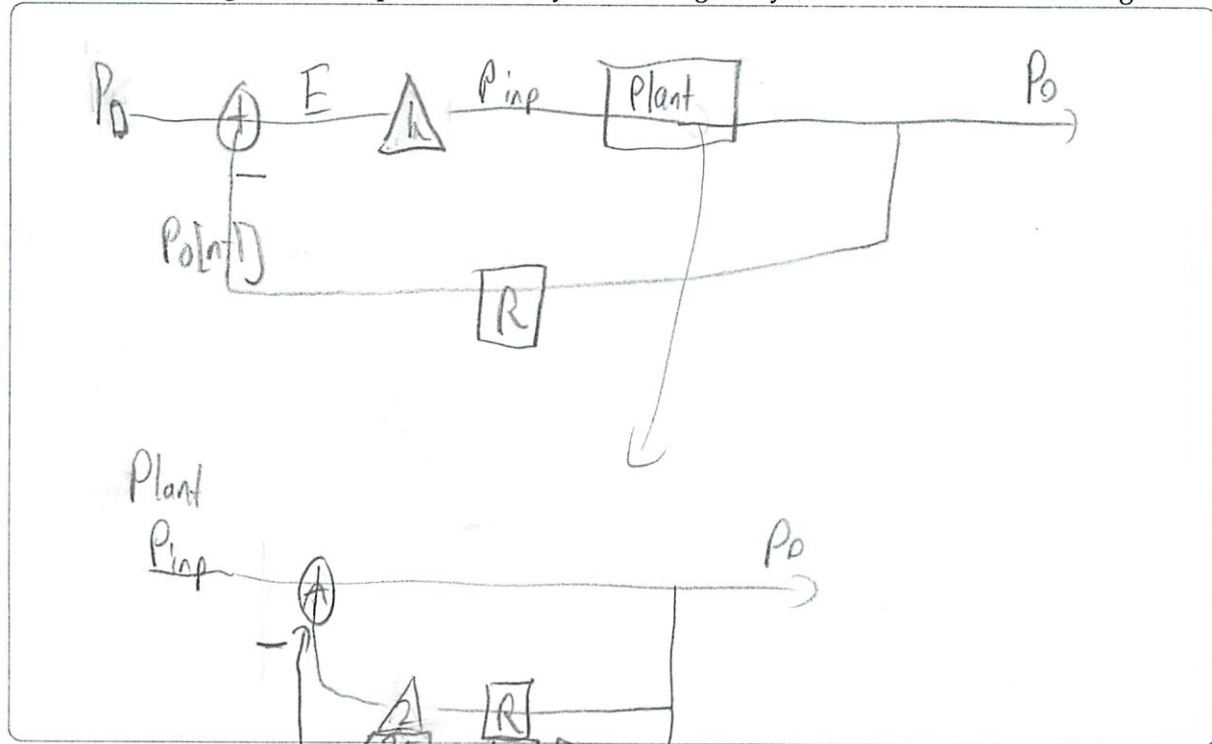
$$P_{inp}[n] = \underset{\substack{\uparrow \\ \text{gain}}}{k} (P_d - P_o[n-1]) + 5$$

$$P_o = ? \quad \text{you have it in part 4 but not here} + 3$$

n	-2	-1	0	1	2	3	4
$P_i[n]$	-	-	-	-	-	-	-
$P_o[n]$	0	0	10	20	$40 - \frac{10}{4}$	$75 - 5$	
					$37\frac{1}{2}$	70	

$$\begin{array}{r} 1 \\ 37 \\ \cdot 2 \\ \hline 74 \end{array}$$

Draw a block diagram that represents this system, using delays, adders/subtractors and gains.



What is the system function that characterizes $\frac{P_o}{P_d}$? Use k to denote the gain in your system.

$$P_o[n] = 2P_o[n-1] - .25P_o[n-2] + k(P_d - P_o[n-1])$$

$$\frac{P_o}{P_d} = ?$$

is old part relevant?

die fast hopefully right

5 Predicting Growth (20 Points)

The following Python classes differ only in the boxed regions.

```
class GrowthA(SM):
    startState = (0,0)
    def getNextValues(self, state, input):
        (s0,s1) = state
        output = input + s0 + 2*s1
        newState = (s1,output)
        return (newState,output)
```

```
class GrowthB(SM):
    startState = (0,0)
    def getNextValues(self, state, input):
        (s0,s1) = state
        output = input + 2*s0 + s1
        newState = (s1,output)
        return (newState,output)
```

```
class GrowthC(SM):
    startState = (0,0)
    def getNextValues(self, state, input):
        (s0,s1) = state
        output = input + s0 + 2*s1
        newState = (s1,input)
        return (newState,output)
```

```
class GrowthD(SM):
    startState = (0,0)
    def getNextValues(self, state, input):
        (s0,s1) = state
        output = input + 2*s0 + s1
        newState = (s1,input)
        return (newState,output)
```

State $(\overset{s_0}{y[n-2]}, \overset{s_1}{y[n-1]})$ top row $A+B$
 state $(x[n-2], x[n-1])$ bottom row $(+D)$

Part a. Determine which (if any) of GrowthA, GrowthB, GrowthC, and GrowthD generate state machines whose output $y[n]$ at time n is given by

$$y[n] = x[n] + x[n-1] + 2x[n-2]$$

for times $n \geq 0$, when the input $x[n] = 0$ for $n < 0$.

Circle all of the classes that satisfy this relation, or circle **none** if none satisfy it.

GrowthA

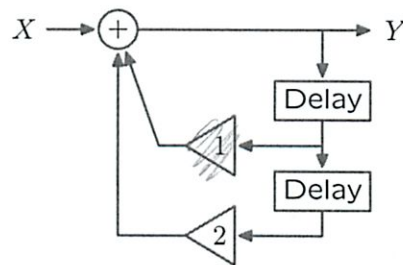
GrowthB

GrowthC

GrowthD

none

Part b. Determine which (if any) of GrowthA, GrowthB, GrowthC, and GrowthD generate state machines whose input-output relation can be expressed as the following block diagram.



Feedback

$$\frac{H_1}{1-H_1H_2}$$

$$H_1 = 1$$

$$H_2 = R$$

Circle all of the classes that satisfy this relation, or circle **none** if none satisfy it.

GrowthA

GrowthB

GrowthC

GrowthD

none

$$\frac{1}{1-1(-R)} = \frac{1}{1+R}$$

$$\frac{1}{1-1(-2R^2)} = \frac{1}{1+2R^2}$$

$$\frac{Y}{X} = \frac{1}{(1+R)(1+2R^2)}$$

well slightly

different

Part c. Let H_A , H_B , H_C , and H_D represent the system functions associated with the state machines generated by GrowthA, GrowthB, GrowthC, and GrowthD, respectively. Fill in the following table to indicate the number and locations of the poles of H_A , H_B , H_C , and H_D . Pole locations can be listed in any order. Leave unnecessary entries blank.

system	# of poles	pole 1 location	pole 2 location	pole 3 location
H_A	2	$1+\sqrt{2}$	$1-\sqrt{2}$	
H_B	2	$\frac{1}{2}$	-1	
H_C	0			
H_D	0			

work
→on back
next 2

pgs

$$H_A / y[n] = x[n] + y[n-2] + 2y[n-1]$$

$$y(1 - 2R - R^2) = x$$

$$\frac{Y}{X} = \frac{1}{-R^2 - 2R + 1}$$

$$\frac{1}{-(\frac{1}{2})^2 - 2(\frac{1}{2}) + 1} \cdot z^2$$

$$\frac{z^2}{-1 - 2z + z^2}$$

$$\frac{z^2}{(x-1)(x-1)}$$

$$\frac{-2 \pm \sqrt{4 - 4 \cdot 1 \cdot -1}}{2 \cdot 1}$$

$$1 \pm \frac{\sqrt{8}}{2}$$

$$1 \pm \frac{\sqrt{2 \cdot 4}}{2} = 1 \pm \sqrt{2}$$

$$H_B / y[n] = x[n] + 2y[n-2] + y[n-1]$$

$$y(1 - R + 2R^2) = x$$

$$\frac{Y}{X} = \frac{1}{-2R^2 - R + 1}$$

$$\frac{1}{-2(\frac{1}{2})^2 - (\frac{1}{2}) + 1} \cdot z^2$$

$$\frac{z^2}{-2 - z + z^2}$$

$$\frac{1 \pm \sqrt{1 - (4 \cdot 1 \cdot -2)}}{2}$$

$$\frac{1 \pm \sqrt{9}}{2}$$

$$\frac{4}{2}$$

$$\frac{-2}{2}$$

$(-1) < \text{dominate}$

6 I need a caffeine jolt (15 Points)

Taking exams is hard work, and it would be nice if there were a caffeine dispenser next to every student's desk. You are to create a state machine that dispenses caffeine jolts (unfortunately these are expensive, and cost 4 dollars each!). This machine accepts dollar bills in different denominations, but does not make change. Hence, your machine should have the following behavior:

- The inputs to the machine are positive integers (representing different denominations of dollars);
- If the input plus the current amount of money deposited in the machine is greater than 4, the machine outputs the current input; *Won't take extra money*
- If the input plus the current amount of money deposited in the machine is exactly 4, the machine outputs a jolt and resets its internal state;
- If the input plus the current amount of money deposited in the machine is less than 4, the machine adjusts its state, outputs None and waits for the next input.

Here are some examples:

```
>>>v = Vending()
>>>v.transduce([1,1,1,1])
[None, None, None, 'jolt']
```

```
>>>v.transduce([1,3])
[None, 'jolt']
```

```
>>>v.transduce([5,1,6,3])
[5, None, 6, 'jolt']
```

Feel free to change the startState if it simplifies your solution. Here is an outline of our state machine:

```
class Vending(sm.SM):
    startState = None
    def getNextValues(self, state, inp):
```

Cost = 4

More States for #5

Hc

$$y[n] = x[n] + x[n-2] + 2x[n-1]$$

$$Y = X + XR^2 + 2XR$$

$$\frac{Y}{X} = \frac{R^2 + 2R + 1}{1}$$

Roots

None

H0

$$y[n] = x[n] + 2x[n-2] + x[n-1]$$

$$Y = X + 2XR^2 + XR$$

$$\frac{Y}{X} = \frac{2R^2 + R + 1}{1}$$

Roots none

(state = money in)

cost = 4

Complete the definition of getNextValues

Start state = None

- 3

```
def getNextValues(self, state, inp):  
    if (state + inp) == cost: # volt  
        return (0, 'volt')  
    elif (state + inp) < cost: # accept $  
        return (state + inp, None)  
    elif (state + inp) > cost: # return money  
        return (state, inp)  
    else:  
        return 'error'
```



6.01 Midterm 1 Solutions: Fall 2010

Name:**Section:**

Enter all answers in the boxes provided.

This solution is not correct for people who took the make-up exam on Wednesday morning starting at 8AM.

During the exam you may:

- read any paper that you want to
- use a calculator

You may not

- use a computer, phone or music player

For staff use:

1.	/12
2.	/16
3.	/16
4.	/21
5.	/20
6.	/15
total:	/100

1 OOP (12 points)

The following definitions have been entered into a Python shell:

```
class Account:
    chargeRate = 0.01
    def __init__(self, start):
        self.value = start
    def debit(self, amount):
        debitAmt = min(amount, self.value)
        self.value = self.value - debitAmt
        return debitAmt
    def deposit(self, amount):
        self.value += amount
    def fee(self, baseAmt):
        self.debit(baseAmt * self.chargeRate)
    def withdraw(self, amount):
        if self.value >= 10000.0:
            self.fee(amount/2.0)
        else:
            self.fee(amount)
        return self.debit(amount)

class Checking(Account):
    chargeRate = 0.05
    def deposit(self, amount):
        if self.value <= 1:
            Account.deposit(self, (1-self.chargeRate) * amount)
        else:
            Account.deposit(self, amount)
```

Assume that the following expressions have been evaluated:

```
Eric = Checking(4000.0)
Ellen = Account(4000.0)
```

Write the values of the following expressions. Write None when there is no value; write Error when an error results and explain briefly why it's an error. Assume that these expressions are evaluated one after another (all of the left column first, then right column).

Eric.withdraw(3000.0)

3000.0

Ellen.withdraw(3000.0)

3000.0

Eric.value

850.0

Ellen.value

970.0

Eric.withdraw(1000.0)

800.0

Ellen.withdraw(1000.0)

960.0

Eric.value

0

Ellen.value

0.0

Eric.deposit(5000.0)

None

Ellen.deposit(5000.0)

None

Eric.value

4750.0

Ellen.value

5000.0

2 So who kicks b*** (16 points)

Here are some class definitions, meant to represent a league of football teams.

```
class Team:
    def __init__(self, name, wins, losses, pointsFor, pointsAgainst):
        self.name = name
        self.wins = wins
        self.losses = losses
        self.pointsFor = pointsFor
        self.pointsAgainst = pointsAgainst

class League:
    def __init__(self):
        self.teams = {}
    def addTeam(self, team):
        self.teams[team.name] = team
    def updateGame(self, teamName, ptsFor, ptsAgin):
        # to be filled in
    def computeStat(self, proc, filt):
        return [proc(team) for team in self.teams.values() if filt(team)]
```

Imagine instantiating these classes by:

```
Pats = Team('Pats', 5, 1, 150, 100)
Ravens = Team('Ravens', 4, 2, 80, 30)
Colts = Team('Colts', 1, 4, 100, 200)
```

```
NFL = League()
NFL.addTeam(Pats)
NFL.addTeam(Ravens)
NFL.addTeam(Colts)
```

We would like to be able to update our information by adding in new game data. For example, if the Pats beat the Colts, by a score of 30 to 15, the record for the Pats should include another win, and an updated record of points for and points against; similarly for the Colts. We would do this by calling

```
NFL.updateGame('Pats', 30, 15)
NFL.updateGame('Colts', 15, 30)
```

Write a Python procedure that will complete the definition of `updateGame`. You may assume that all teams exist in the instance of the league, and that there are no ties, only wins and losses. Please make sure that the indentation of your written solution is clear.

```
def updateGame(self, teamName, ptsFor, ptsAgin):  
    team = self.teams[teamName]  
    if ptsFor > ptsAgin:  
        team.wins += 1  
    else:  
        team.losses += 1  
    team.pointsFor += ptsFor  
    team.pointsAgainst += ptsAgin
```

Assume that the NFL has been defined as above, but with more teams. Write an expression using `computeStat`, to be evaluated by the Python interpreter, that will return a list of wins for all teams in the NFL. (It is okay to define and use helper functions.) For the example instance defined above, your expression should return the list:

[5, 4, 1]

```
NFL.computeStat(lambda y: y.wins, lambda x: True)
```

Write an expression using `computeStat`, to be evaluated by the Python interpreter, that will return a list of the losses for the Pats and the Colts, where each entry includes the name of the team. (It is okay to define and use helper functions.) For the example instance defined above, your expression should return the list:

[['Pats', 1], ['Colts', 4]]

```
NFL.computeStat(lambda y: [y.name, y.losses],  
                 lambda x: x.name == 'Pats' or x.name == 'Colts')
```


3 Will it or won't it? (16 Points)

For each difference equation below, say whether, for a unit sample input signal:

- the output of the system it describes will diverge or not as n approaches infinity, assuming that $x[n], y[n] = 0$ for $n < 0$,
- the output of the system it describes (a) will always be positive, (b) will alternate between positive and negative, or (c) will have a different pattern of oscillation

Part 1:

$$5y[n] + 2y[n-1] = x[n-2]$$

diverge? Yes or No

No

Why?

root's magnitude less than 1

positive/alternate/oscillate

Alternate

Why?

root's sign is negative

Part 2:

$$y[n] = -2y[n-1] - 5y[n-2] + x[n-1]$$

diverge? Yes or No

Yes

Why?

largest root's magnitude greater than 1

positive/alternate/oscillate

Oscillates

Why?

pole is complex

4 Grow, baby, grow (21 Points)

You and your colleague in the Biology Department are growing cells. In each time period, every cell in the bioreactor divides to yield itself and one new daughter cell. However, due to aging, half of the cells die after reproducing (don't worry about the details of how accurately this models real cell division).

We can describe this system with the following difference equation. We let P_o denote the number of cells at each time step.

Then

$$P_o[n] = 2P_o[n-1] - 0.5P_o[n-2]$$

Suppose that $P_o[0] = 10$ and $P_o[n] = 0$ if $n < 0$. What are the first few values for the number of cells (note that while not physically realistic, our model might provide fractional answers)?

$$P_o[0] = 10$$

$$P_o[1] = 20$$

$$P_o[2] = 35$$

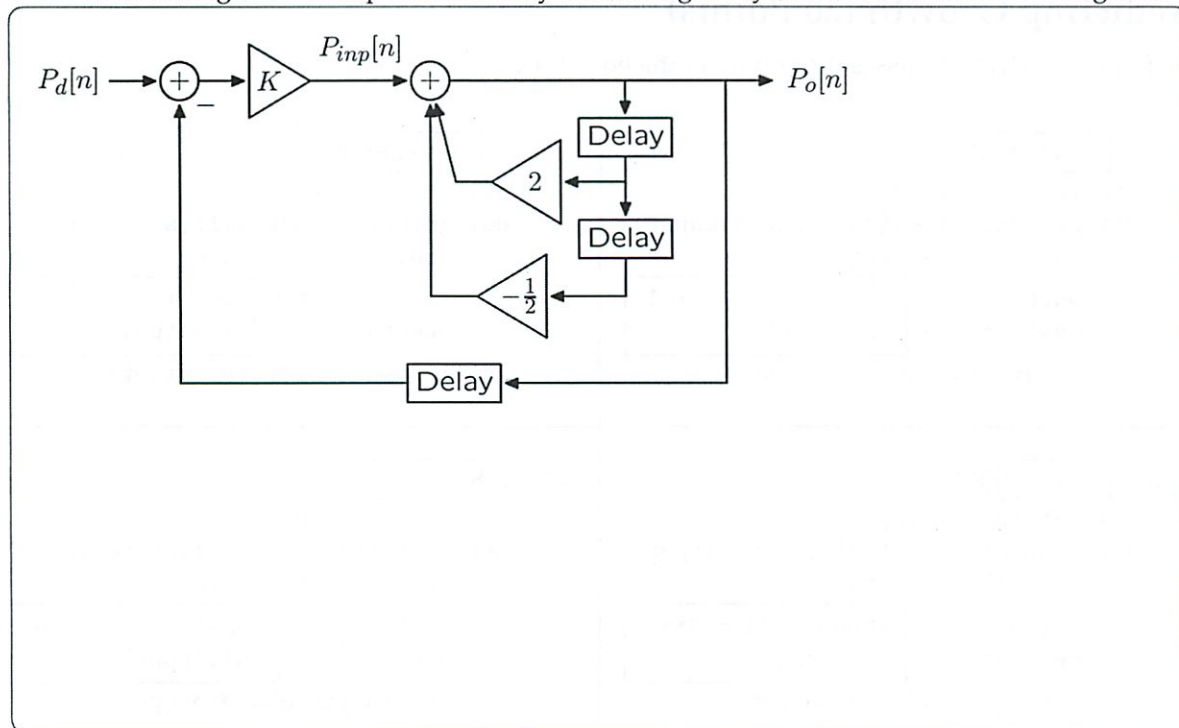
$$P_o[3] = 60$$

Your goal is to create a constant population of cells, that is, to keep P_o constant at some desired level P_d . You are to design a proportional controller that can add or remove cells as a function of the difference between the actual and desired number of cells. Assume that any additions or deletions at time n are based on the measured number of cells at time $n-1$. Denote the number of cells added or removed at each step P_{inp} . Derive the difference equations that govern this closed loop system.

$$P_o[n] = 2P_o[n-1] - .5P_o[n-2] + P_{inp}[n]$$

$$P_{inp}[n] = k(P_d[n] - P_o[n-1])$$

Draw a block diagram that represents this system, using delays, adders/subtractors and gains.



What is the system function that characterizes $\frac{P_o}{P_d}$? Use k to denote the gain in your system.

$$\frac{k}{0.5R^2 + (k-2)R + 1}$$

5 Predicting Growth (20 Points)

The following Python classes differ only in the boxed regions.

```
class GrowthA(SM):
    startState = (0,0)
    def getNextValues(self, state, input):
        (s0,s1) = state
        output = input + s0 + 2*s1
        newState = (s1,output)
        return (newState,output)
```

```
class GrowthB(SM):
    startState = (0,0)
    def getNextValues(self, state, input):
        (s0,s1) = state
        output = input + 2*s0 + s1
        newState = (s1,output)
        return (newState,output)
```

```
class GrowthC(SM):
    startState = (0,0)
    def getNextValues(self, state, input):
        (s0,s1) = state
        output = input + s0 + 2*s1
        newState = (s1,input)
        return (newState,output)
```

```
class GrowthD(SM):
    startState = (0,0)
    def getNextValues(self, state, input):
        (s0,s1) = state
        output = input + 2*s0 + s1
        newState = (s1,input)
        return (newState,output)
```

Part a. Determine which (if any) of GrowthA, GrowthB, GrowthC, and GrowthD generate state machines whose output $y[n]$ at time n is given by

$$y[n] = x[n] + x[n-1] + 2x[n-2]$$

for times $n \geq 0$, when the input $x[n] = 0$ for $n < 0$.

Circle all of the classes that satisfy this relation, or circle **none** if none satisfy it.

GrowthA

GrowthB

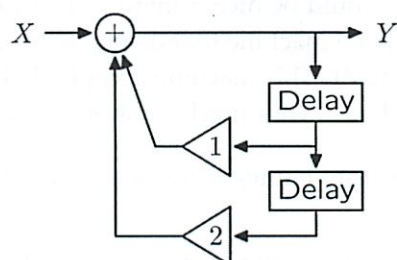
GrowthC

GrowthD

none

GrowthD

Part b. Determine which (if any) of GrowthA, GrowthB, GrowthC, and GrowthD generate state machines whose input-output relation can be expressed as the following block diagram.



Circle all of the classes that satisfy this relation, or circle **none** if none satisfy it.

GrowthA

GrowthB

GrowthC

GrowthD

none

GrowthB

Part c. Let H_A , H_B , H_C , and H_D represent the system functions associated with the state machines generated by GrowthA, GrowthB, GrowthC, and GrowthD, respectively. Fill in the following table to indicate the number and locations of the poles of H_A , H_B , H_C , and H_D . Pole locations can be listed in any order. Leave unnecessary entries blank.

system	# of poles	pole 1 location	pole 2 location	pole 3 location
H_A				
H_B				
H_C				
H_D				

2, $1 + \sqrt{2}$, $1 - \sqrt{2}$

2, 2, -1

0

0

6 I need a caffeine jolt (15 Points)

Taking exams is hard work, and it would be nice if there were a caffeine dispenser next to every student's desk. You are to create a state machine that dispenses caffeine jolts (unfortunately these are expensive, and cost 3 dollars each!). This machine accepts dollar bills in different denominations, but does not make change. Hence, your machine should have the following behavior:

- The inputs to the machine are positive integers (representing different denominations of dollars);
- If the input plus the current amount of money deposited in the machine is greater than 3, the machine outputs the current input;
- If the input plus the current amount of money deposited in the machine is exactly 3, the machine outputs a jolt and resets its internal state;
- If the input plus the current amount of money deposited in the machine is less than 3, the machine adjusts its state, outputs None and waits for the next input.

Here are some examples:

```
>>>v = Vending()
>>>v.transduce([1,1,1])
[None, None, 'jolt']
```

```
>>>v.transduce([1,2])
[None, 'jolt']
```

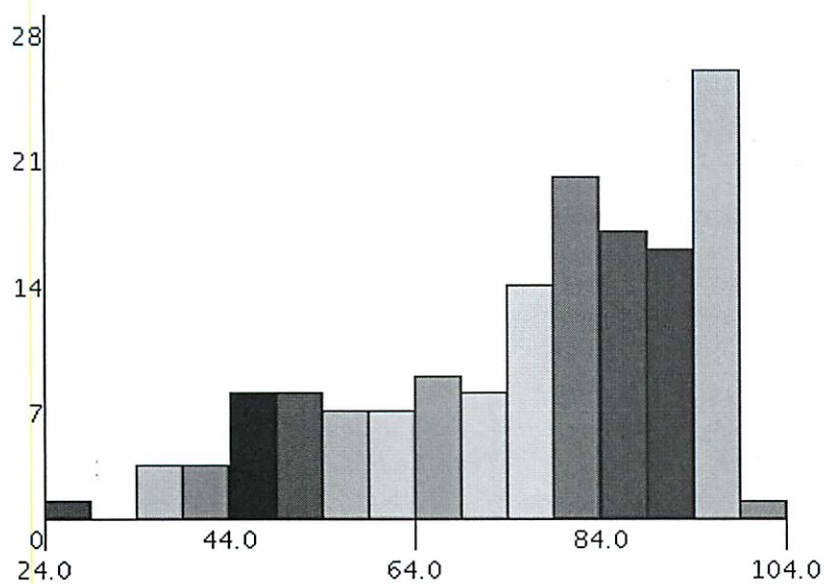
```
>>>v.transduce([5,1,4,2])
[5, None, 4, 'jolt']
```

Feel free to change the `startState` if it simplifies your solution. Here is an outline of our state machine:

```
class Vending(sm.SM):
    startState = None
    def getNextValues(self, state, inp):
```


Complete the definition of getNextValues

```
class Vending(sm.SM):
    startState = 0    # Note the choice of startState
    def getNextValues(self, state, inp):
        if state + inp > 3:
            return (state, inp)
        elif state + inp == 3:
            return (None, 'jolt')
        else:
            return (state + inp, None)
```



10/12

6.01: Introduction to EECS I

Designing Control Systems

Week 6

October 12, 2010

Outline

- Complex poles
- Designing control systems

Reading: Chapter 6

Midterm exam:

- Tonight! 7:30–9:00 PM
- 32-141 or 32-155
- Any printed material okay
- No computers or phones
-
- No software lab today!

From last time

- Behavior of a system can be captured by its system function, which characterizes the relationship between input and output
- System functions can be combined just as PCAP modules can be combined
- Primitives are gains and delays
- Combinations include cascades, positive feedback and negative feedback
- Behavior of system captured by poles of system

*Make design choices to get systems how you want

Poles: Summary

- The **poles** of a system are the roots of the denominator polynomial of the system function in $1/z$.
- The **dominant pole** is the pole with the largest magnitude.

Correction

Dependence on pole magnitude

Response to a bounded input signal, if the dominant pole has magnitude

- > 1 : output signal may be unbounded
- < 1 : output signal will be bounded
if the input is transient, output signal will converge to 0.
- 1 : output signal may be bounded

A system is *stable* if the dominant pole has magnitude less than 1, i.e. if all the poles are inside the unit circle.

Example: accumulator

Dependence on pole type

Response to a transient input signal, if the dominant pole is

- **real and positive**: output signal will, after finitely many steps, begin to increase or decrease monotonically.
- **real and negative**: output signal will, after finitely many steps, begin to alternate signs.
- **complex**: output signal will, after finitely many steps, begin to be periodic, with a period of $2\pi/\Omega$, where Ω is the 'angle' of the pole in the complex plane.

Complex Roots

What if a root has a non-zero imaginary part? *i or j*

Factor theorem: express a polynomial as a product of factors, with one factor associated with each root of the polynomial.

Fundamental theorem of algebra: a polynomial of order n has n roots. The roots can have imaginary parts.

$$\frac{Y}{X} = \frac{b_0 + b_1\mathcal{R} + b_2\mathcal{R}^2 + b_3\mathcal{R}^3 + \dots}{1 + a_1\mathcal{R} + a_2\mathcal{R}^2 + a_3\mathcal{R}^3 + \dots}$$

Factor denominator:

$$\frac{Y}{X} = \frac{b_0 + b_1\mathcal{R} + b_2\mathcal{R}^2 + b_3\mathcal{R}^3 + \dots}{(1 - p_0\mathcal{R})(1 - p_1\mathcal{R})(1 - p_2\mathcal{R})(1 - p_3\mathcal{R}) \dots}$$

Partial fractions:

$$\frac{Y}{X} = \frac{C_0}{1 - p_0\mathcal{R}} + \frac{C_1}{1 - p_1\mathcal{R}} + \frac{C_2}{1 - p_2\mathcal{R}} + \dots + D_0 + D_1\mathcal{R} + D_2\mathcal{R}^2 + \dots$$

How does a mode from a complex root behave?

can write as factor (1 -)(1 -)

Complex Poles

Difference equations that represent physical systems (e.g., population growth, bank accounts, etc.) have real-valued coefficients.

Bank account with interest:

$$y[n] = (1 + r)y[n-1] + x[n]$$

Wall Finder:

$$d_o[n] = d_o[n-1] + KTd_o[n-2] - KTd_i[n-1]$$

Difference equations with real-valued coefficients generate real-valued outputs from real-valued inputs.

But they might still have complex poles.

Representing complex numbers

- Start with $p = a + jb$

- Convert to polar coordinates r, Ω

$$r = \sqrt{a^2 + b^2}; \quad \Omega = \tan^{-1}(b/a)$$

- Euler's definition of complex exponents:

$$re^{j\Omega} = r(1 + j\Omega + \frac{(j\Omega)^2}{2!} + \frac{(j\Omega)^3}{3!} + \frac{(j\Omega)^4}{4!} + \dots)$$

$$= r((1 - \frac{\Omega^2}{2!} + \frac{\Omega^4}{4!} - \dots) + j(\Omega - \frac{\Omega^3}{3!} + \frac{\Omega^5}{5!} - \dots))$$

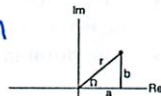
$$= r(\cos \Omega + j \sin \Omega)$$

$$= r(\frac{a}{\sqrt{a^2 + b^2}} + j \frac{b}{\sqrt{a^2 + b^2}})$$

$$= a + jb$$

This representation makes exponentiation easy:

$$(re^{j\Omega})^n = r^n e^{jn\Omega}$$



*Even terms come out
signs alternate*

Complex Poles

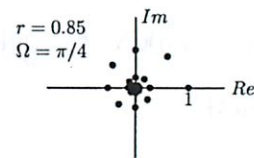
Complex-valued poles produce complex-valued modes.

Because modes are geometric series with ratio $p\mathcal{R}$,

$$\frac{1}{1 - p\mathcal{R}} = 1 + p\mathcal{R} + p^2\mathcal{R}^2 + \dots + p^n\mathcal{R}^n + \dots$$

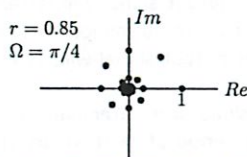
It is convenient to express the base p of a complex-valued mode in polar form. Let $p = re^{j\Omega}$. Then

$$\frac{1}{1 - re^{j\Omega}\mathcal{R}} = 1 + re^{j\Omega}\mathcal{R} + r^2e^{j2\Omega}\mathcal{R}^2 + \dots$$



Complex Poles

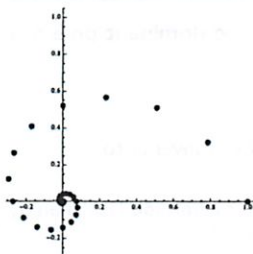
$$\frac{1}{1 - re^{j\Omega}\mathcal{R}} = 1 + re^{j\Omega}\mathcal{R} + r^2e^{j2\Omega}\mathcal{R}^2 + \dots$$



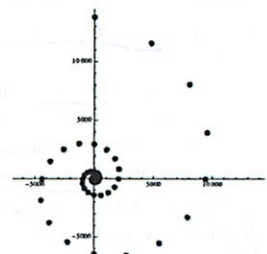
- Magnitude of samples is geometric sequence with ratio $r = |p|$. (In other words, magnitude "multiplies" on each time step.)
- Phase angle of samples grows linearly with time, slope = Ω . (In other words, angles "adds" on each time step.)
- Period of signal is $2\pi/\Omega$.

magnitude is dominating

Convergence and Divergence



$$p = 0.85e^{j\pi/8} \approx 0.785 + 0.325j$$



$$p = 1.1e^{j\pi/8} \approx 1.016 + 0.421j$$

Complex Roots

An isolated complex root can result only from a difference equation with complex-valued coefficients.

Example:

$$\frac{Y}{X} = \frac{1}{1 - re^{j\Omega}\mathcal{R}}$$

Corresponding difference equation:

$$y[n] - re^{j\Omega}y[n-1] = x[n]$$

Complex Roots

If p is a root of a polynomial with constant real-valued coefficients, then its complex conjugate p^* is also a root.

Proof. Let $D(z)$ represent a polynomial in z with constant real-valued coefficients.

If p is a root of $D(z)$ then $D(p) = 0$.

Since all of the coefficients are real-valued,

$$D(p^*) = (D(p))^* = 0^* = 0.$$

Thus p^* is also a root.

need odd # of terms

Complex Roots

If we pair the factors corresponding to complex-conjugate roots, the resulting polynomial has real-valued coefficients.

$$\frac{Y}{X} = \frac{1}{(1 - re^{j\Omega}\mathcal{R})(1 - re^{-j\Omega}\mathcal{R})} = \frac{1}{1 - 2r \cos \Omega \mathcal{R} + r^2 \mathcal{R}^2}$$

Note that

$$(1 - re^{j\Omega}\mathcal{R})(1 - re^{-j\Omega}\mathcal{R}) = 1 - r(e^{j\Omega} + e^{-j\Omega})\mathcal{R} + r^2 \mathcal{R}^2$$

Recall that

$$e^{jx} = \cos x + j \sin x$$

and that

$$\sin(-x) = -\sin(x) \quad \cos(-x) = \cos(x)$$

So

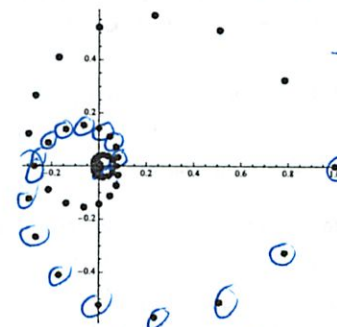
$$(e^{j\Omega} + e^{-j\Omega}) = 2 \cos \Omega$$

Complex modes, Real results

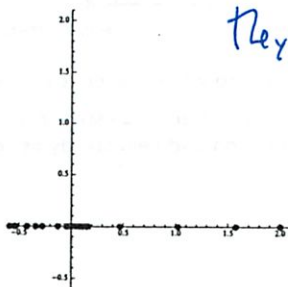
$$p = 0.85e^{j\pi/8} \approx 0.785 + 0.325j$$

$$p^* = 0.85e^{-j\pi/8} \approx 0.785 - 0.325j$$

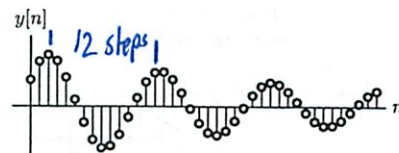
conjugate also is a root just flipped across real axis

**Complex modes, Real results**

$$\begin{aligned} p^n + (p^*)^n &= (0.85)^n e^{j(n\pi/8)} + (0.85)^n e^{-j(n\pi/8)} \\ &= (0.85)^n [e^{j(n\pi/8)} + e^{-j(n\pi/8)}] \\ &= 2 * (0.85)^n * \cos n\pi/8 \end{aligned}$$

**Check Yourself**

Output of a system with poles at $z = re^{\pm j\Omega}$.



Which of the following are true?

- $r < 0.5$ and $\Omega \approx 0.5$
- $0.5 < r < 1$ and $\Omega \approx 0.5$
- $r < 0.5$ and $\Omega \approx 0.08$
- $0.5 < r < 1$ and $\Omega \approx 0.08$
- none of the above

$$\text{period} = \frac{2\pi}{\Omega} = \frac{2\pi}{\# \text{ steps}} = 5$$

fairly close to 1

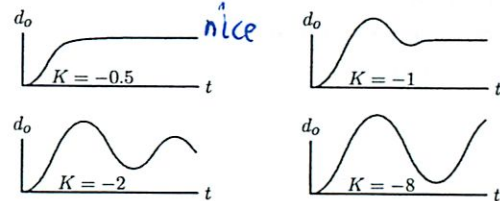
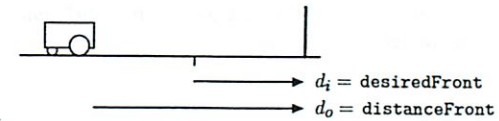
*-denom in system function is key
-output is still real - since comes in pairs*

Designing a Control System

Exploring different system designs: we can sometimes pick

- delays
- gains
- time constants

Wall Finder



What causes these different types of responses?
Is there a systematic way to optimize K ?

what is the best gain to use?

Wall Finder with instant sensor



proportional controller: $v[n] = Ke[n] = K(d_i[n] - d_s[n])$

locomotion: $d_o[n] = d_o[n-1] - Tv[n-1]$

sensor with no delay: $d_s[n] = d_o[n]$

The difference equations provide a concise description of behavior.

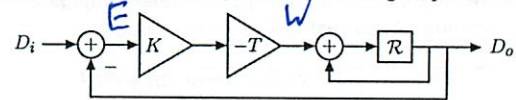
$$d_o[n] = d_o[n-1] - Tv[n-1] = d_o[n-1] - TK(d_i[n-1] - d_o[n-1])$$

However it provides little insight into how to choose the gain K .

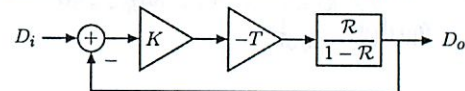
hard to know what K to use

Wall Finder with instant sensor

We can represent the entire system with a single system function.



Replace accumulator with equivalent block diagram.



Equivalent system with a single block:

$$D_i \rightarrow \frac{-KTR}{1 - (1 + KT)R} \rightarrow D_o$$

Modular! But we still need a way to choose K .

$$\begin{aligned} W &= -KT \\ E &= D_i - D_o \\ D_o &= R(W + D_o) \\ &= R(-KT E + D_o) \\ &= R(-KT(D_i - D_o) + D_o) \\ &= RD_o - KTR(D_i - D_o) \end{aligned}$$

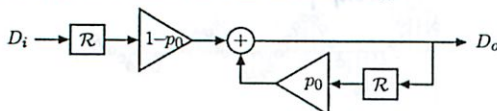
Wall Finder with instant sensor

The system function contains a single pole at $z = 1 + KT$.

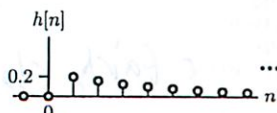
$$\frac{D_o}{D_i} = \frac{-KTR}{1 - (1 + KT)R}$$

The numerator is just a gain and a delay.

The whole system is equivalent to the following:



where $p_0 = 1 + KT$. Here is the unit sample response for $KT = -0.2$:



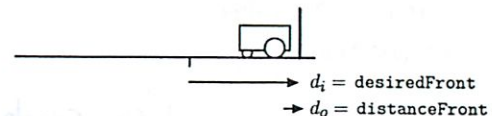
slowly decaying

to unit impulse

-but don't do that in real life

Wall Finder with instant sensor

We are often interested in the step response of a control system.

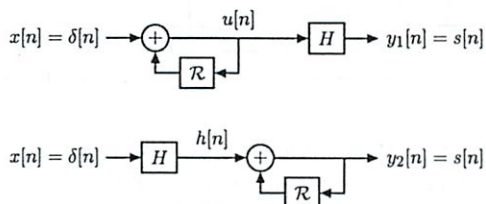


Start the output D_o at zero while the input is held constant at one.

Said another way, we are often interested in the system response when the input is turned on and held steady at some value.

Step Response

The response of a system (represented by H) to the unit step signal is equal to the accumulated responses to the unit sample signal.



$y_1[n] = y_2[n]$ because these systems are commutative (provided each starts at rest).

Thus, if we know the unit sample response of our system, we can just accumulate (or integrate) to get unit step response!

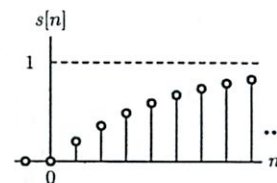
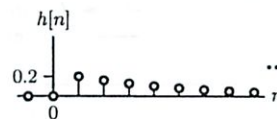
how quickly will it ramp up to system

Wall Finder with instant sensor

System step response slow because unit sample response slow.

$$\frac{D_o}{D_i} = \frac{-KTR}{1 - (1 + KT)R} = \frac{(1 - p_0)R}{1 - p_0R}; \quad p_0 = 1 + KT$$

Initial response is $1 - p_0$, each subsequent response scaled by p_0 .



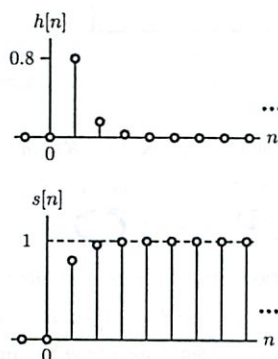
if hold the system on (input)

add up each piece - accumulate

Wall Finder with instant sensor

Step response faster if $KT = -0.8$ (i.e., $p_0 = 0.2$).

Initial response is $1 - p_0$, each subsequent response scaled by p_0 .

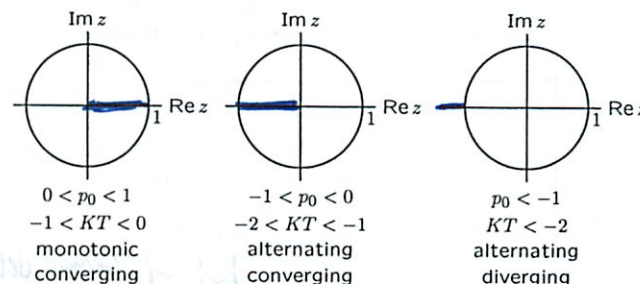


k bigger in magnitude - faster convergence

Wall Finder with instant sensor

The poles of the system function provide insight for choosing K .

$$\frac{D_o}{D_i} = \frac{-KTR}{1 - (1 + KT)R} = \frac{(1 - p_0)R}{1 - p_0R}; \quad p_0 = 1 + KT$$



Check Yourself

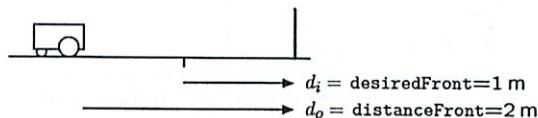
Find KT for fastest convergence of unit sample response.

$$\frac{D_o}{D_i} = \frac{-KTR}{1 - (1 + KT)R}$$

1. $KT = -2$
2. $KT = -1$
3. $KT = 0$
4. $KT = 1$
5. $KT = 2$
0. none of the above

Wall Finder with instant sensor

The optimum gain K moves robot to desired position in one step.



$$KT = -1$$

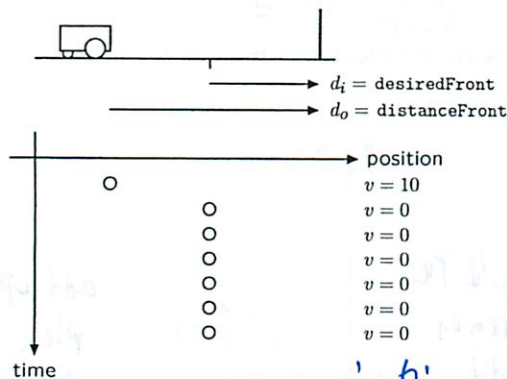
$$K = -\frac{1}{T} = -\frac{1}{1/10} = -10$$

$$v[n] = K(d_i[n] - d_o[n]) = -10(1 - 2) = 10 \text{ m/s}$$

exactly the right speed to get there in one step!

Wall Finder with instant sensor

The optimum gain K moves robot to desired position in one step.

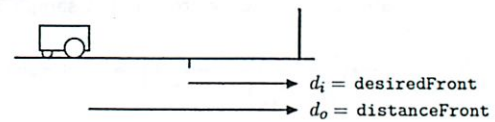


is this realistic?

- momentum
- sensor has delay

Analysis of Wall Finder System: Adding Sensory Delay

Adding delay tends to destabilize control systems.



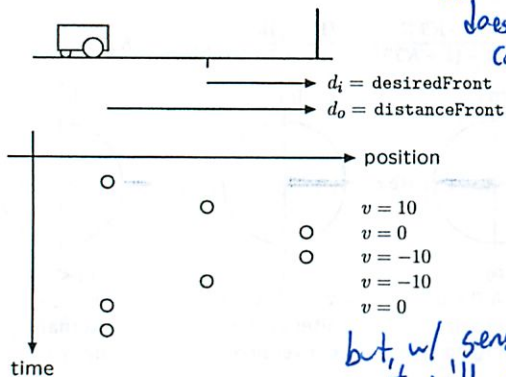
proportional controller: $v[n] = Ke[n] = K(d_i[n] - d_s[n])$

locomotion: $d_o[n] = d_o[n-1] - Tv[n-1]$

sensor with delay: $d_s[n] = d_o[n-1]$

Analysis of Wall Finder System: Adding Sensory Delay

Adding delay tends to destabilize control systems.

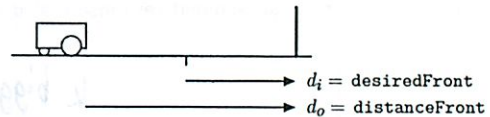


*assuming
does not
crash*

*but, w/ sensor delay
it will oscillate*

Analysis of Wall Finder System: Adding Sensory Delay

Adding delay tends to destabilize control systems.



proportional controller: $v[n] = Ke[n] = K(d_i[n] - d_s[n])$

locomotion: $d_o[n] = d_o[n-1] - Tv[n-1]$

sensor with delay: $d_s[n] = d_o[n-1]$

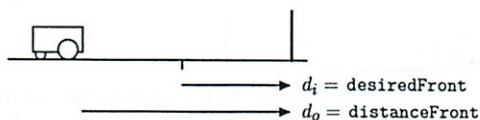
The difference equations provide a concise description of behavior.

$d_o[n] = d_o[n-1] - Tv[n-1] = d_o[n-1] - TK(d_i[n-1] - d_o[n-2])$

However it provides little insight into how to choose the gain K .

Analysis of Wall Finder System: Block Diagram

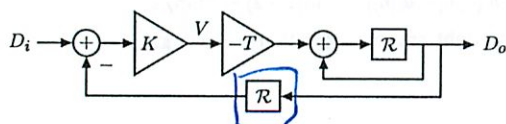
Incorporating sensor delay in block diagram.



proportional controller: $v[n] = Ke[n] = K(d_i[n] - d_s[n])$

locomotion: $d_o[n] = d_o[n-1] - Tv[n-1]$

sensor with delay: $d_s[n] = d_o[n-1]$

**Analyzing wallFinder: Poles**

Substitute $\mathcal{R} \rightarrow \frac{1}{z}$ in the system function to find the poles.

$$\frac{D_o}{D_i} = \frac{-KTR}{1 - \mathcal{R} - KTR^2} = \frac{-KT\frac{1}{z}}{1 - \frac{1}{z} - KT\frac{1}{z^2}} = \frac{-KTz}{z^2 - z - KT}$$

find pole

The poles are then the roots of the denominator.

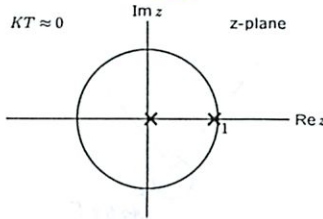
$$z = \frac{1}{2} \pm \sqrt{\left(\frac{1}{2}\right)^2 + KT}$$

new poles

Feedback and Control: Poles

If KT is small, the poles are at $z \approx 0$ and $z \approx 1$.

$$z = \frac{1}{2} \pm \sqrt{\left(\frac{1}{2}\right)^2 + KT} \approx \frac{1}{2} \pm \sqrt{\left(\frac{1}{2}\right)^2} = 0, 1$$



Pole near 0 generates fast response.

Pole near 1 generates slow response.

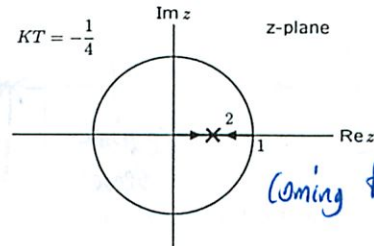
Slow mode (pole near 1) dominates the response.

↳ biggest magnitude

Feedback and Control: Poles

As KT becomes more negative, the poles move toward each other and collide at $z = \frac{1}{2}$ when $KT = -\frac{1}{4}$.

$$z = \frac{1}{2} \pm \sqrt{\left(\frac{1}{2}\right)^2 + KT} = \frac{1}{2} \pm \sqrt{\left(\frac{1}{2}\right)^2 - \frac{1}{4}} = \frac{1}{2}, \frac{1}{2}$$



Coming together

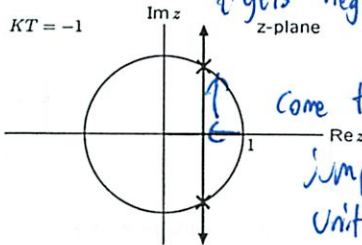
Persistent responses decay. The system is stable.

↳ half is best you can do

Feedback and Control: Poles

If $KT < -1/4$, the poles are complex.

$$z = \frac{1}{2} \pm \sqrt{\left(\frac{1}{2}\right)^2 + KT} = \frac{1}{2} \pm j\sqrt{-KT - \left(\frac{1}{2}\right)^2}$$



gets negative

come together

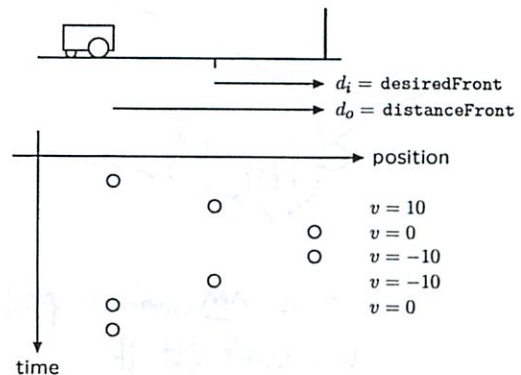
jump up to unit circle

Complex poles → oscillations.

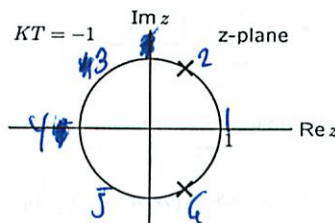
- if try and do better than $KT = -1/4$ then you have oscillations

Same oscillation we saw earlier!

Adding delay tends to destabilize control systems.



Check Yourself

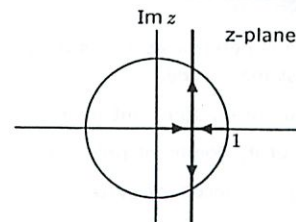


What is the period of the oscillation?

1. 1 2. 2 3. 3
4. 4 5. 6 0. none of above

Time steps till it comes back around

Check Yourself



closed-loop poles

$$\frac{1}{2} \pm \sqrt{\left(\frac{1}{2}\right)^2 + KT}$$

Find KT for fastest response.

1. 0 2. $-\frac{1}{4}$ 3. $-\frac{1}{2}$
4. -1 5. $-\infty$ 0. none of above

best response

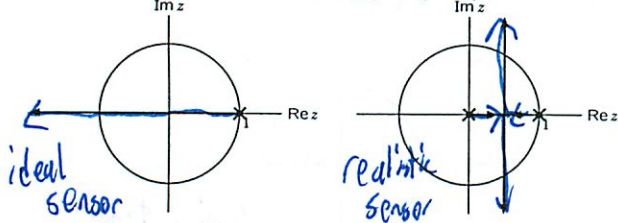
7 want fastest possible response
magnitude of dominate pole as little as possible

Destabilizing Effect of Delay

Adding delay in the feedback loop makes it more difficult to stabilize.

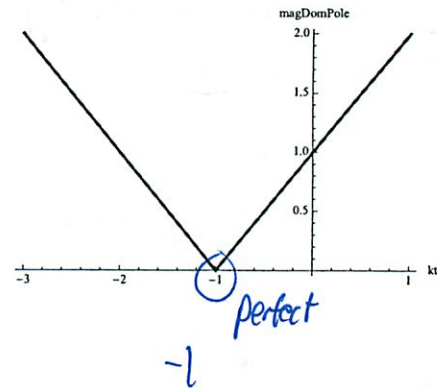
Ideal sensor: $d_s[n] = d_o[n]$

More realistic sensor (with delay): $d_s[n] = d_o[n-1]$

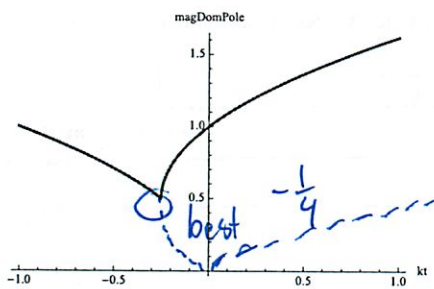


Fastest response without delay: single pole at $z = 0$.

Fastest response with delay: double pole at $z = \frac{1}{2}$. much slower!

Magnitude of pole versus kt : no sensor delay

is there python code to do this?

Magnitude of dominant pole versus kt : with sensor delay

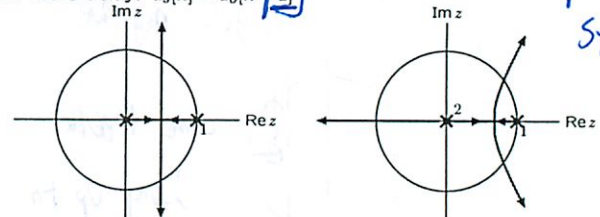
-- is non dominate poll
- we can't get it

Destabilizing Effect of Delay

Adding more delay in the feedback loop is even worse.

More realistic sensor (with delay): $d_s[n] = d_o[n-1]$

Even more delay: $d_s[n] = d_o[n-2]$



Fastest response with delay: double pole at $z = \frac{1}{2}$.

Fastest response with more delay: double pole at $z = 0.682$.

→ even slower

slow down
response of
system

independent of input

Designing Control Systems: Summary

System Functions provide a convenient summary of information that is important for designing control systems.

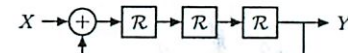
The long-term response of a system is determined by its dominant pole — i.e., the pole with the largest magnitude.

A system is unstable if the magnitude of its dominant pole is > 1 .

A system is stable if the magnitude of its dominant pole is < 1 .

Delays tend to decrease the stability of a feedback system.

Check Yourself



How many of the following statements are true? 4

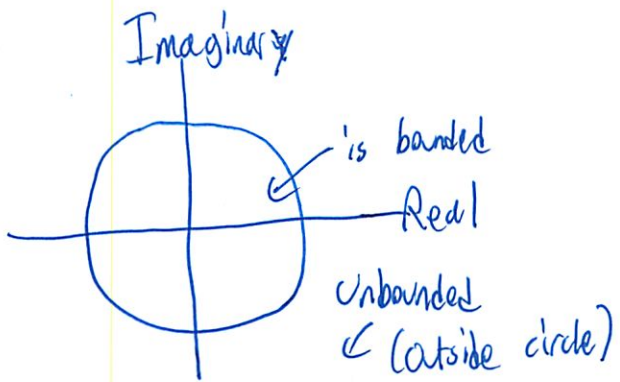
1. This system has 3 poles. *denom is cubic*
2. Unit sample response is the sum of 3 geometric sequences.
3. Unit-sample response is $y[n] : 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, \dots$
4. Unit-sample response is $y[n] : 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, \dots$
5. One of the poles is at $z = 1$.

Can be written as product of 3 linear terms

~~XXXXXXXXXXXX~~

$$Y = R^3(Y + X)$$

$$Y(1 - R^3) = R^3 X$$



Pole Position

2 checks

which is which

~~not~~

b - alternate signs + bounded

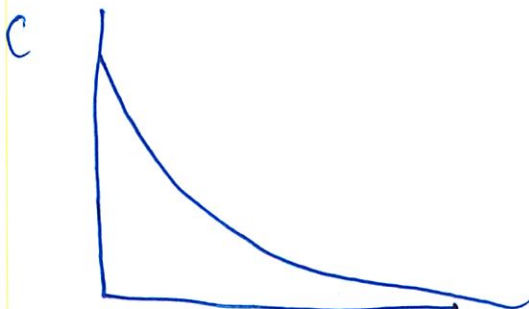
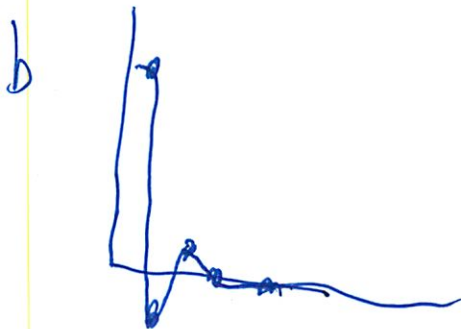
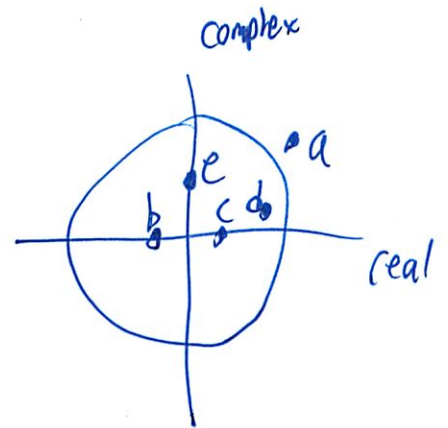
a = complex alt sign

c - bounded monotonic

e = guessing over compensate

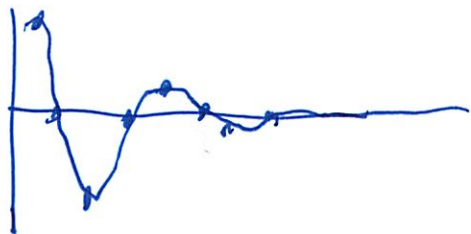
- kinda guessing

e x \rightarrow so flip
b x

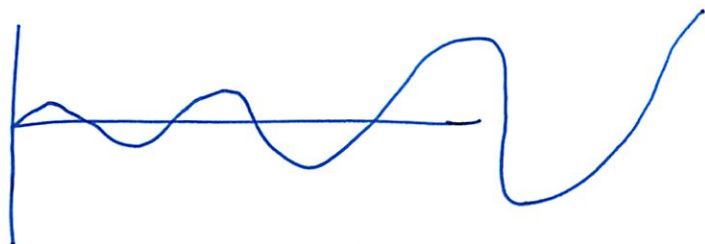


(2)

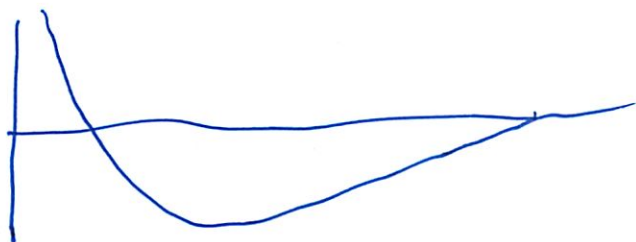
e



a



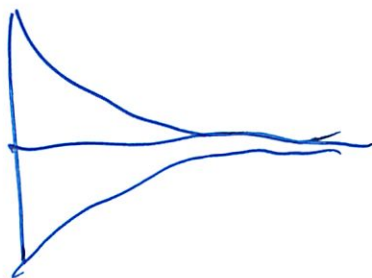
d



e has ^{oscillation} cycle of 4 ^{period} - since $\frac{1}{4}$ around circle
- fine steps

magnitude ratio of envelope is dying

$\sqrt{x^2 + y^2}$
distance from center

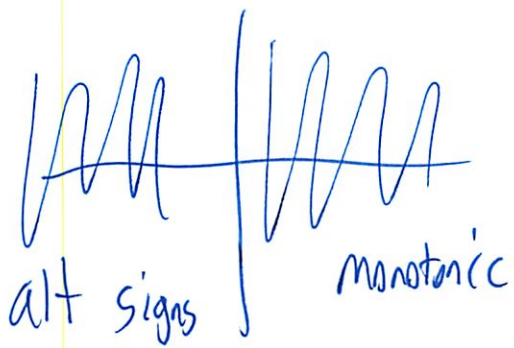


$$\frac{2}{3} \quad \frac{3}{2} \quad \frac{4}{3}$$

b has period 2

c period 1

③



Design Lab 6: Sizable Following

1 Introduction

This lab should be done with a partner. Each partnership should have

- a lab laptop
- a robot.

Do athrun 6.01 update to get the files for this lab, which will be in Desktop/6.01/lab6/designLab/.

The relevant files in the distribution are:

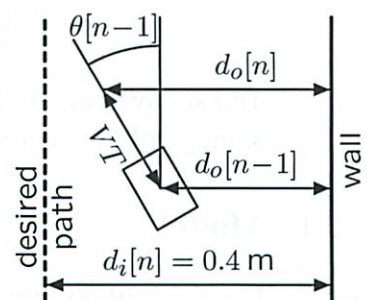
- `delayPlusPropBrainSkeleton.py`: A brain with a place for you to write the delay-plus-proportional controller.
- `anglePlusPropBrainSkeleton.py`: A brain with a place for you to write the angle-plus-proportional controller.
- `wallTestWorld.py`: A world with a wall for the robot to follow. (This file is in the worlds subdirectory.)
- `dl6Work.py`: A file with appropriate imports for making system functions and for optimization.

Be sure to mail all of your code and plots to your partner. Each of you will need to bring copies with you to your first interview.

Last week, you wrote a proportional controller to move the robot parallel to a wall while trying to maintain a constant, desired distance from the wall. The forward velocity V was set to a constant (0.1 m/s) and the angular velocity $\omega[n]$ was proportional to the error signal $e[n]$, which was the difference between the desired distance $d_i[n]$ and current distance $d_o[n]$.

It is important to note that the distances d_i , d_o and d_s are all distances from the **center** of the robot to the wall on its right.

Unfortunately, no value of the proportionality constant k gave good performance. Large values of k gave fast oscillations and small values of k gave large errors, especially when the initial angle of the robot was not parallel to the wall. In this lab, we will develop two new types of controllers to achieve better performance.



The steps in this lab are:

- Make a controller that depends on the previous distance to the wall as well as the current one:
 - Build a model of the delay-plus-proportional controller-plant-sensor system, using the `SystemFunction` class.
 - Use the model to find the best gains.
 - Build a corresponding controller for the robot.
 - Test it in simulation for the various gains.
 - Test it on a real robot.
- Make a controller that depends on the current angle to the wall, as well as the current distance:
 - Build a model of the angle-plus-proportional controller-plant-sensor system, using the `SystemFunction` class.
 - Use the model to find the best gains.
 - Build a corresponding controller for the robot.
 - Test it in simulation for the various gains.
 - Test it on a real robot.

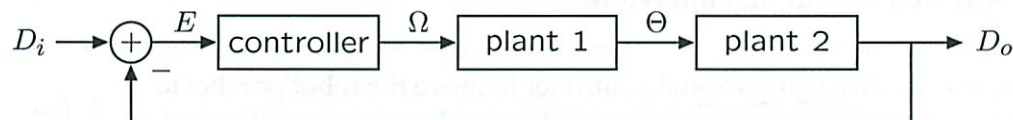
2 Remembrance of Things Past

We can make a better wall-following controller by processing the error signal E in a more sophisticated way than we did in the previous lab. For example, we could adjust the angular velocity using some combination of the present and previous values of the error,

$$\omega[n] = k_1 e[n] + k_2 e[n-1].$$

did not use before
error

We refer to this controller as “delay plus proportional.” Notice that the system has the same form as the one from last week.



The subsystems that represent robot locomotion (e.g. plant 1 and plant 2) and the sensor are the same. Only the controller has changed.

2.1 Model

- Step 1.** Use the Python `SystemFunction` class to model and analyze the behavior of whole system when the robot has a delay-plus-proportional controller, as follows.

- Open the file `dl6Work.py` (which imports `sf`) in Idle and use it to do the work in this part of the lab.
- Write a Python procedure `delayPlusPropModel`, that takes gains `k1` and `k2` as input and returns a `SystemFunction` that describes the behavior of the system when the robot has a delay-plus-proportional controller. You can assume that $T = 0.1$ seconds and $V = 0.1$ m/s.

Notice that we have added new combinators: `sf.FeedforwardAdd` and `sf.FeedforwardSubtract`. You may also use the other `sf` combinators listed in the **Software Documentation** section of the **Reference Material** from the course web page.

You may find your answer to last week's problem [Wk.5.3.3](#) helpful.

Wk.6.1.1. Part 1 Enter your `delayPlusPropModel` code into the tutor.

2.2 Picking gains

We want to pick the values of `k1` and `k2` to get the best stable behavior. As we saw in lecture, behavior is improved by reducing the magnitude of the dominant pole. Let's consider the case of a single gain first, the system you modeled last week. You could construct a function $f(k)$ that computes the magnitude of the system's dominant pole. Now, you want to find the value of k that produces the minimum value of this function.

Given a function $f(x)$, how can we find a value x^* such that $f(x^*) \leq f(x)$ for all x ? If f is differentiable, then we can do this relatively easily by taking the derivative, setting it to 0 and solving for x . This gets tricky when the function f is complicated, when there may be multiple minima, and/or when we wish to extend to functions with multiple arguments. For functions that aren't differentiable (such as those involving `max` or `abs`), there is no straightforward mathematical approach at all. In one dimension, if we know a range of values of x that is likely to contain the minimum, we can plausibly sample different values of x in that range, evaluate f at each of them, and return the sampled x for which $f(x)$ is minimized.

The function `optimize.optOverLine` does this. It is called as follows:

```
optimize.optOverLine(objective, xmin, xmax, numXsteps, compare)
```

- `objective`: a procedure that takes a single numeric argument and returns a numeric value,
- `xmin`, `xmax`: a range of values for the argument,
- `numXsteps`: how many points to test within the range,
- `compare`: an optional comparison function that defaults to be `operator.lt`, which is the less than, `<`, operator in Python. This means that if you don't specify the `compare` argument, the procedure will return the value that **minimizes** the objective.

This function returns a tuple (bestObjValue, bestX) consisting of the best value of the objective and the x value that corresponds to it.

Step 2. Consider four different values of k_1 : 10, 30, 100, and 300. For each value of k_1 , use `optimize.optOverLine` to determine the value of k_2 that minimizes the magnitude of the least stable pole.

Be sure to test both positive and negative values of k_2 , be sure that you have tested a large enough range, and be sure that, ultimately, you have sampled at a granularity of at least 0.1. Rather than setting up one long minimization run with a wide range and a small granularity, it's better to start with a coarse granularity to find the right rough value, and then search more finely around that. Hint: the magnitude of k_2 should not be bigger than that of k_1 .

We encourage you to define and use the bestk2 procedure stub that is included in `dl6Work.py` for this purpose.

k_1	k_2	magnitude of dominant pole
10		
30		
100		
300		

Wk.6.1.1 Part 2 Enter the values of k_2 and the magnitude of the associated dominant pole that you found for each of the values of k_1 above.

2.3 Brain

Step 3. Implement the delay plus proportional controller by editing the `WallFollower` state machine class in `delayPlusPropBrainSkeleton.py`. Think very carefully about what you want to output in the first time step.

As before, the brain has two parts connected in cascade. The first part is an instance of the `Sensor` class, which implements a state machine whose input is a sequence of instances of `SensorInputs` and whose output is the perpendicular distance to the wall. The perpendicular distance is calculated by `getDistanceRight` in the `sonarDist` module by using triangulation (assuming the wall

Sensor
machine +
wall follower

same as before?

sensor

is locally straight). If sensors 6 and 7 both hit the wall, then the value is fairly accurate. If only one of them hits it, then it's less accurate. If neither sensor hits the wall, then it returns 1.5. The code for the Sensor class is provided.

The second part of the brain is an instance of the WallFollower class. You should provide code so that the WallFollower class implements a state machine whose input is the perpendicular distance to the wall and whose output is an instance of the class `io.Action`. Think carefully about what you are going to store in the state of the machine, and how you will initialize `startState`.

The brain is set up so that whenever you click **Stop**, a plot will be displayed, showing how the perpendicular distance to the right wall changes as a function of time. To save this plot, take a screen shot, as described on our web page. **This plot will disappear once you reload the brain.**

Step 4.

Run your brain in the world `wallTestWorld.py` (in the `worlds` directory). Determine how the behavior of the system is affected by the controller gains k_1, k_2 . Use only the gains that you determined in the previous step. Pay attention to the distance values being printed out by the brain. Save plots to illustrate the performance of each of your optimized gain pairs k_1, k_2 . Choose names for these files so that you can remember the parameters that were used to generate each one. Keep the files for your oral interview.

Don't change your controller to try to make it work better! Instead, try to understand the different kinds of failures that can happen and how they depend on the choice of gain or initial condition.

Check Yourself 1. Which of the four gain pairs work best in simulation?

$k_1 =$

$k_2 =$

Which gains cause bad behavior?

Step 5. Go to the edge of the room or out to the hallway to find a long (at least two bubble-wrapped boards) wall to work with.

Run your brain on a real robot. Get a measuring stick and be sure to start the robot at the same initial conditions (0.5 meters from the wall and rotated $\pi/8$ radians to the left) as in the simulator. Pay attention to the distance values being printed out by the brain. Save a plot for each of your calculated gain pairs. Keep the files for your oral interview.

What are the different states - which controller to use

why? not 5

Check Yourself 2. Which of the four gain pairs work best on a robot?

$k_1 =$

$k_2 =$

Are the best gains the same as in simulation?

Which gains cause bad behavior?

Checkoff 1.

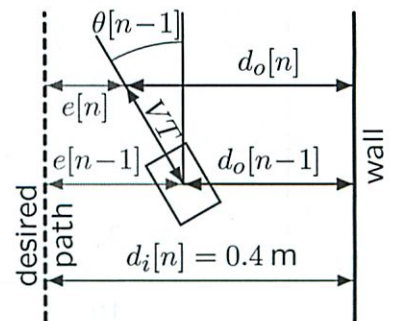
Show a staff member plots for the simulated and real robot runs, and discuss their relationship. How is the robot's behavior related to the magnitude of the dominant pole, for each of the gain pairs?

3 If we are facing in the right direction, all we have to do is keep on walking

Why does the delay-plus-proportional controller from the previous section behave better than the proportional controller from last week? The only difference is access to $e[n-1]$. So why should old information be helpful? A related issue is why the best values of k_2 were just a bit larger than those for $-k_1$.

A better way to think about the delay-plus-proportional controller is as proportional-plus-derivative, where the derivative here is the first difference. As we will see below, if we re-parameterize the gains in terms of differences instead of delays, then the relation between the gains ($k_2 = -k_1 + \epsilon$) is no longer mysterious.

For this particular problem, the difference $e[n] - e[n-1]$ can be interpreted graphically in terms of the angle $\theta[n-1]$ (see right figure). Thus the delay-plus-proportional controller can base the next angular velocity (its output) on both the position AND angle of the robot. Notice however that the information about position is for time n while the information about angle is for time $n-1$.



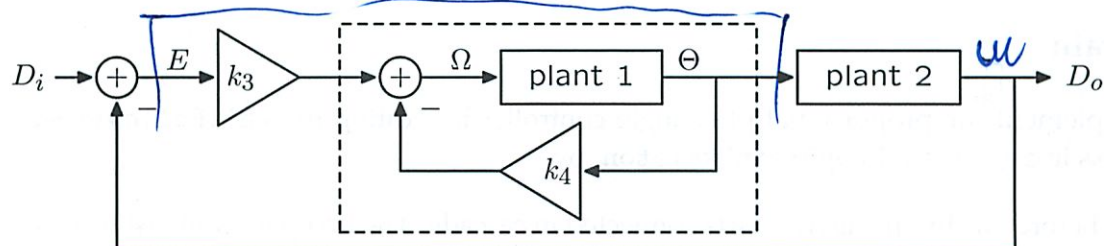
Our robot has more than just one sensor – it actually has eight sonars arranged at slightly different angles – so we can measure the angle directly. How well could a control system work if it had up-to-date information about both position and angle? We can answer this question by analyzing an angle-plus-proportional controller:

$$\omega[n] = k_3(d_i[n] - d_o[n]) + k_4(\theta_d - \theta[n])$$

where $\theta_d = 0$ represents the desired angle (as measured relative to the wall), so that

$$\omega[n] = k_3 e[n] - k_4 \theta[n]$$

as shown in the following block diagram.



Notice that this controller has a feedback loop to control the angle Θ that is *inside* a second feedback loop to control the distance D_o .

3.1 Model

- Step 6.** In `dl6Work.py`, write a Python procedure `anglePlusPropModel` that takes gains `k3` and `k4` as input and returns a `SystemFunction` that describes the system with angle-plus-proportional control. Assume that $T = 0.1$ seconds and $V = 0.1$ m/s.

Wk.6.1.3 Part 1 Enter your `anglePlusPropModel` code into the tutor.

- Step 7.** For `k3` equal to 1, 3, 10, and 30, determine values of `k4` that minimize the magnitude of the least stable pole of the angle-plus-proportional system.

k3	k4	magnitude of dominant pole
1		
3		
10		
30		

Wk.6.1.3 Part 2 Enter the values of `k4` and the magnitude of the associated dominant pole that you found for each of the values of `k3` above.

3.2 Brain

- Step 8.** Implement the proportional plus angle controller by editing the `WallFollower` state machine class in `anglePlusPropBrainSkeleton.py`.

As before, the brain has two parts connected in cascade. The first part is an instance of the `Sensor` class, which implements a state machine whose input is a sequence of instances of `SensorInputs` and whose output is a sequence of pairs of the perpendicular distance to the wall on the right and the angle to the wall.

The second part of the brain is an instance of the `WallFollower` class. You should provide code so that the `WallFollower` class implements a state machine whose input is a **pair of the perpendicular distance to the wall on the right and the angle to the wall** and whose output is an instance of the class `io.Action`.

Notice that if sonar 6 or 7 is out of range, then we cannot calculate the angle, and the second component of the output of the Sensor machine will be `None`. When that happens, your brain should set the angular velocity to 0.

- Step 9.** Test that your brain works in the soar simulator with the `wallTestWorld.py` world. Save a plot for each of your calculated gain pairs. Keep the files for your oral interview.

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

$k_3 =$

$k_4 =$

Which gains cause bad behavior?

- Step 10.** Go to the edge of the room or out to the hallway to find a long (at least two bubble-wrapped boards) wall to work with.

Run your brain on a real robot. Get a measuring stick and be sure to start the robot at the same initial conditions (0.5 meters from the wall and rotated $\pi/8$ radians to the left) as in the simulator. Pay attention to the distance values being printed out by the brain. Save a plot for each of your calculated gain pairs. Keep the files for your oral interview.

Check Yourself 4. Which of the four gain pairs work best on a robot?

$k_3 =$

$k_4 =$

Are the best gains the same as in simulation?

Which gains cause bad behavior?

Checkoff 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?

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

des Lab 6

15/18

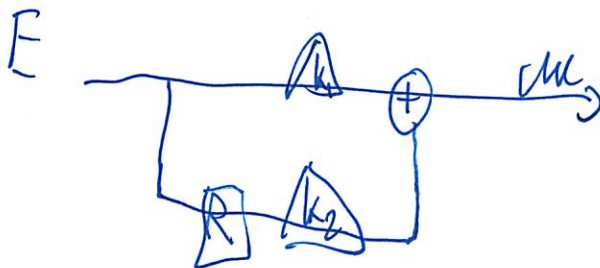
- only need to define a new controller
- plant 1, 2, cascade same
- old controller

sf. Gain(k)

but this does not do error input



- new controller



have we done feedforward like this?

- yeah, but not recently

sf. Feedforward Add (sf. gain(k), sf. Cascade(sf. R(), sf. Gain(k2)))

②

So in this case they are not functions
just say controller
- as variable

Pt2 Gains

k_1 k_2 ? \swarrow which k_2 is lowest pole for given k_1
magnitude dom Pole

10

30

100

300

- best k_2 function

Optimize. opt over line

- what ~~the~~ ~~the~~ objective?

~~delay~~

lambda x: delay Plus Prop Model (k_1 , x)

? spelled lambda

what k_2 min + max?

+/- large range

1

-300, 300, x1 $\frac{600}{1}$

③

What does it return

- Should be Tuple (best obj Value, best X)

- getting SF and x value

— should return a single numeric value

abs((k1, x), ~~best~~ dominate Pole())

So ^{best k1} 10, -300, 300, 6000) ← takes a while to run

1995, -9.9 (✓)
↑ ↑
dominate Value of
pole k2
mag

now k1 = 30

(19847, -29.7)

k1 = 100

(19456, -97.3)

k = 300

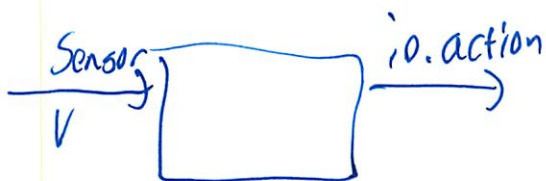
.7722, -271.7

④

* want magnitude as small as possible
- high k_1 , high k_2

2.3 Brain

- Sensor is done
- wall following get next value



but where does the u output from controller come in?

Can we ref it in the sm?

- same as last week

$$u = \text{error} \cdot k_1 + \text{error} \cdot k_2$$

↑
last time error
def state

What should start state be?

0? - the last time

last time was wild

1000 - not going to go anywhere?
- now and in future

60 redo



kinda followed

30

Corrected fast
following wall - dead on
wobbling



← picture is bad
- looked better in real life

100


Big turn in beginning
Then over corrected
lots of wobble

300

almost crashed into wall



waving like a drunk driver

⑤ try 0 now and run on gear
- runs in circle - 

~~try start state~~

try $k_1 = 10$ $k_2 = -9.9$

follows the wall perfectly!

TA: try 300 $k_2 = -279$ w/

start of desired Right

- crazy in beginning, but fixed itself and is now
going good

- need to try all 4 gain controllers

30 crashed!

100 works kinda well

- works best?

Real Robot

10/

aimed out then back in



⑦ 10 redo

actually worked pretty well

* Much slower to react
big correction curve

initial condition problem on 300

- i don't move on 1st try

- start state = None

- if state == None

don't move

read sensor

- that fixed the problem

- need to calculate theta later ← only on subsequent steps

- or type error even though don't use

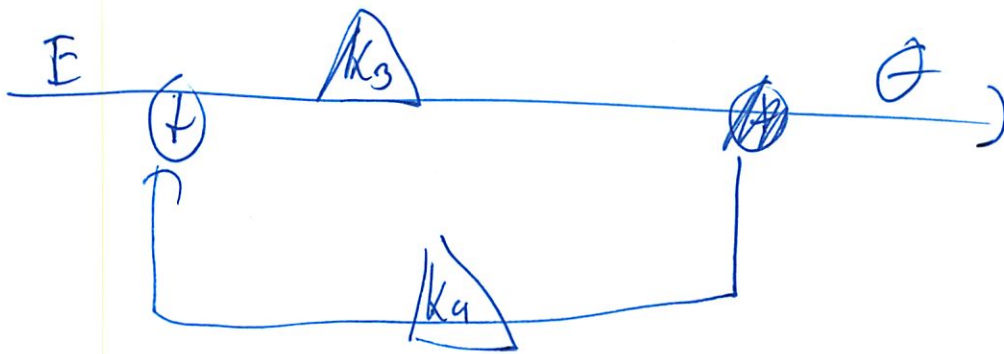
Plant 1 = angle math

feed back the angle from plant 2

— why no controller in code

- part of plant 1

8



Need Θ error

$$\Theta_E = \Theta_d - \Theta[n]$$

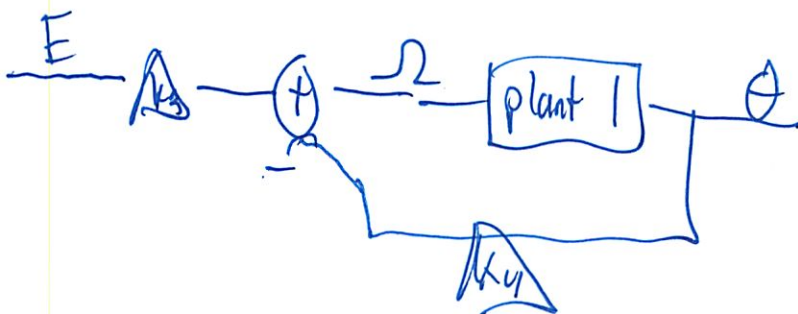
↑ what we measure is $\Theta[n-1]$

$$\Theta_E[n-1] = \Theta_d - \Theta[n-1]$$

-eq they gave us are for plant 2?

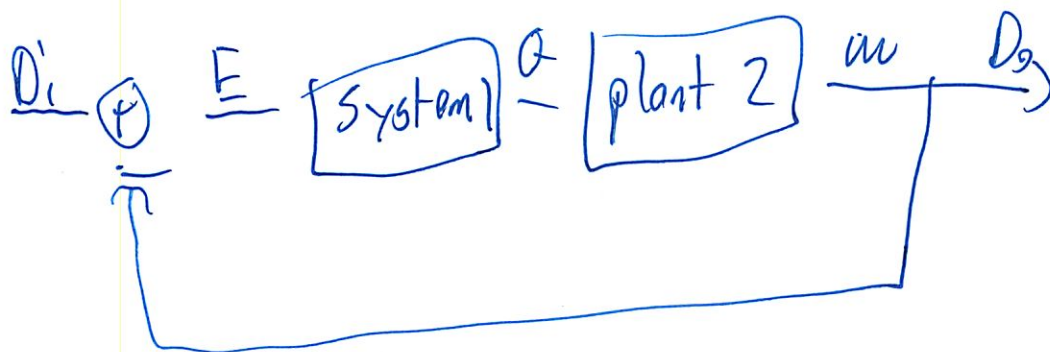
they E K3

Oh just do what they have already



st. Cascade(gain(K3), feedback subtract(plant 1, K4))

⑨ Now combine all of it



sf. feedback subtract (sf. cascade (sys 1, plant 2))

Now best k_4

k_3	k_4	<u>mag dom pole</u>	
1	-1.9 (X)	1.99	(✓)
3	-2.9 (X)	1.99	(✓)
10	-9.9 (X)	1.99	(X)
36	-29.7 (X)	1.98	(X)

all large values

- so bad controller

- or we are wrong

Oh I was using old model

- grr

(10)

<u>k3</u>	<u>k4</u>	<u>mag dom pole</u>
1	1.2 6	.97
3	1.2 1.1	.94
10	2	.90
30	3.43	.83

↑ so gets kinda accurate

Code sensor

- output pair of perp distances distance
to wall and angle

$(dist, \theta)$

if $\theta = \text{None}$

- angular velocity = 0

so need to code ψ

What is θ desired?

90°

degrees, radians?

test if out

- radians

10/16
at home

11

So have dist error + angle error

- how to deal w/ theta

$$W[n] = k_3 \underbrace{[d_i[n] - d_o[n]]}_{\text{error}[0]} + k_4 \underbrace{(\theta_i[n] - \theta[n])}_{\text{error}[1]}$$

and don't need a special start state^{↑ was state last time}

- yeah we don't depend on state now
- but it happens abt it can't calc ~~the~~ theta
then it just goes straight

- try values for k_4

- why does it crash all the time?

- 'if error theta too big'

- when it gets less than ^{theta} 35?

- So I did a bunch of trial + error + it crashes
when using $k_4 * \text{errorTheta}$

↑ calculates

- calculates angle from wall

(12)

flmm calculates ~~on~~ k_4 error theta

problem is adding the two together

- what if print add

- don't use it

- works

? Or $-k_4$?

- circles now before crashing!

- no k_4 ^{program}

What is case where soar will crash?

Or no adding the two

how about $k_3 e[n] - k_4 \Theta[n]$

? ha that worked!

So what was wrong w/ my code ???

See what G.O.I people say via email

now making graphs of each ~~the~~ k value

30 had best slime trail

- well right dist vs step

- but it turned sharply at 1st

$$k_3 = 30$$

$$k_4 = 3.43$$

now need to wait for lab for robot - save plots

Homework 2: Keeping Warm

To get a work file that imports a set of useful packages, download the `hw2.zip` file from the web, unpack `hw2work.py` from it, and put `hw2work.py` in your `lab6/designLab` directory.

Please turn in a single, printed, stapled, clearly readable document with all your answers, including any code, plots, system diagrams or mathematical derivations that support your arguments. Hand it in to 34-501, before the beginning of your nano-quiz on **October 28**.

Introduction

In this problem, we are going to try to understand a system involving a furnace (which itself has a simple control system) and a human who is adjusting the temperature-setting knob on the front of the furnace.

We are going to ask you to make some claims and prove them true, either theoretically or empirically:

- Theoretical tools that you have available are analyzing particular systems through determining their poles, and finding optimal gains by analytically calculating optima of functions.
- Empirical tools that you have available are generating plots of the behavior of particular systems given initial conditions, and finding good gains by systematically sampling a set of candidate gains, evaluating them, and returning the best one.

Demonstrate the correctness of each of your answers below, either empirically or theoretically. Include any code, output, graphs, or mathematical derivations that are necessary to prove your points.

1 The furnace alone

Assume that:

- Input to the furnace is a desired temperature, or 'set point'.
- The controller in the furnace is a simple proportional controller, based on the difference between the set point and the sensed temperature.
- The temperature sensor introduces a one-step delay.

- The behavior of the furnace and the room are such that the temperature is **changed** by an amount equal to the product of the length of time between system updates and the command generated by the controller in the previous time step.
- Throughout this problem, you can just let the length of time between commands be 1.

- Step 1.** Write the difference or operator equations and the system function for the system very clearly. Think about what constitutes the input and output to the system, in order to determine what system function you are deriving.
- Step 2.** Draw the system diagram for the system. Be sure that it matches your derived system function elements.
- Step 3.** Find the gain of the controller that makes the furnace react as quickly as possible to set-point changes while causing the temperature to converge to the desired set point without oscillation. Explain. Remember that you can use the function we employed in lab to optimize an objective function.
- Step 4.** Plot the response of the furnace with your chosen gain to a unit step input. This will model the behavior of the system when the room starts at a temperature 0 and the furnace's set-point rises, at time 0, to a constant value of 1. Here is a handy procedure that takes a system function as input and generates a plot of the behavior resulting from the system starting at rest (all inputs and outputs before time 0 have value 0) with a step signal as input.

```
def plotOutput(sfModel):
    smModel = sfModel.differenceEquation().stateMachine()
    outSig = ts.TransducedSignal(sig.StepSignal(), smModel)
    outSig.plot()
```

2 Human in the loop

Now, imagine an impatient human. The human has a desired temperature, X , and has purchased a furnace with a controller of the type we designed in the previous section. But the human is impatient, and is not content to set the knob on the thermostat to X . Instead, the human continuously adjusts the thermostat, moving the thermostat knob by an increment that is proportional to the difference between X and the sensed temperature of the room. Assume the human has a one-step delay in sensing the temperature of the room, but that his actions have an immediate effect on the knob.

So, if the human thinks the room is too cold—that is, that the sensed temperature, S , is less than X —then he will adjust the set-point up by an amount proportional to $X - S$. Remember to think of the actual knob on the thermostat as an accumulator: its output value is the sum of the adjustments that the human has made to it over time.

- Step 5.** Write the difference or operator equations and system function for the combined human-furnace system very clearly.

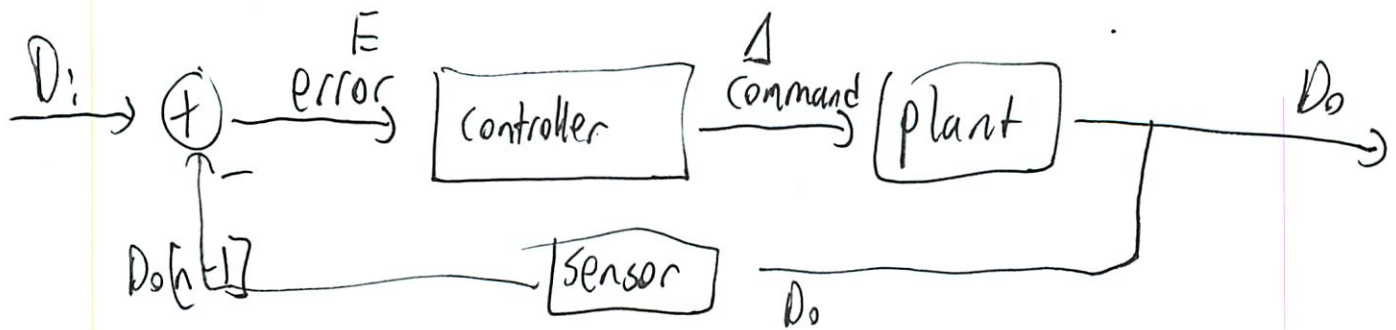
- Step 6.** Draw the system diagram for the system.
- Step 7.** To be sure your system is appropriately formulated, plot the response of the combined system for some values of the gains (remember you have a gain for the human and a gain for the furnace) to a unit step input. This will model the behavior of the system when the room starts at a temperature of 0 and the human's desired temperature rises, at time 0, to a constant value of 1. Be sure the output values make sense.
- Step 8.** Consider what happens when you combine the furnace and controller system (using the best gain you found from part 1) with a human who has gain 1. In general, we'll say the human has gain k if he or she moves the knob an amount equal to k times the difference between the desired temperature and the sensed temperature. Show a simulation and explain.
- Step 9.** If the furnace company knew that it would be delivering furnaces to humans who operated with a gain of 1, would it be able to change the gain in its controller so that the combined human + furnace + controller system operates stably? What gain should the furnace use? Explain. (Remember that you can use either theoretical or empirical tools to help answer these questions).
- Step 10.** If the furnace company knew that it would be delivering furnaces to humans who operated with a gain of 0.5, would it be able to change the gain in its controller so that the combined human + furnace + controller system operates stably? What gain should the furnace use? Explain.
- Step 11.** Is there any gain for the human that causes the furnace (with the gain you found in part 1) to reach its set-point faster, using such a proportional knob-adjustment rule? What gain should the human use? Explain.
- Step 12.** What is the best pairing of human and furnace controller? What gain should the human use and what gain should be set in the furnace? Does this combination do better or worse than just using the furnace on its own? (Note, to find the answer to this question, you may find it useful to optimize a function of two variables. Just as `optimize.optOverLine` takes a procedure of one arguments, a minimum value and maximum value for that argument, and a number of samples within that range, and returns the optimal value, and the argument that yields it, the function `optimize.optOverGrid` applies to a function of two arguments, and takes a minimum value, maximum value and number of steps for each argument, and returns the optimal value and the arguments that yield it.)

Doing HW Assign 2

10/16

No tutor hw this week!

So this is a simple proportional controller like desLab 5



Controller

$$\Delta = k E$$

$$E = D_i - D_o[n-1]$$

← what is a controller again?
just changes input

Plant

$$D_o[n] = \cancel{D_i} \cancel{D_o[n-1]} + \Delta R$$



Sensor

$$D_o[n-1] = D_o R$$

$$\frac{\Delta}{E} = k$$

$$\cancel{D_o} = D_o[n-1] + \Delta$$

$$D_o[1-R] = \Delta$$

$$\frac{D_o}{\Delta} = \frac{1R}{1-R}$$

②

$$\frac{D_0[n-1]}{D_0} = R$$

✓ ? right for sensor
wish I made clean diagram

Multiply for cascade

$$\frac{kR}{1-R}$$

And blocks for Feedback subtract

$$\frac{\frac{kR}{1-R}}{1 - \frac{kR}{1-R} \cdot \frac{A}{1}}$$

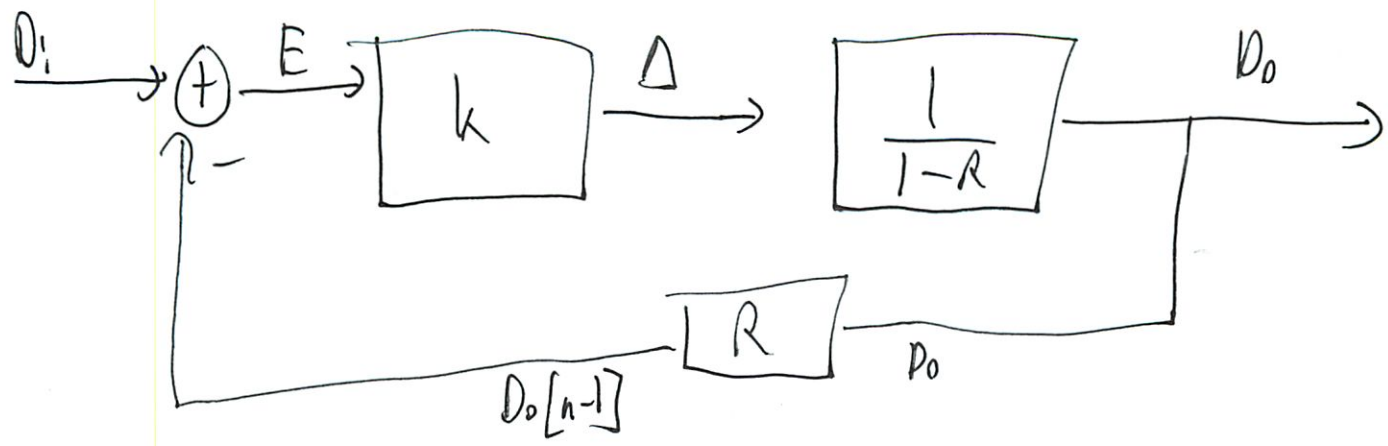
$$\frac{\frac{k}{1-R}}{1 - \frac{kR}{1-R}}$$

$$\frac{k}{1-R} \cdot \left(\frac{1}{1} - \frac{1-R}{kR} \right)$$

$$\frac{k(1-R)}{(1-R)(1-kR)} = \frac{-kR}{1-R-kR+kR^2} = \frac{-kR}{kR^2 + (-k-1)R + 1}$$

No intermediate checks in this lab :-

③



Now find the best gain
Solve for poles w/ $R = \frac{1}{z^2}$

$$\frac{-k\left(\frac{1}{z^2}\right)}{k\left(\frac{1}{z^2}\right)^2 + (-k-1)\left(\frac{1}{z}\right) + 1} \cdot \cancel{z^2} \cdot z^2$$

$$\frac{-k}{k + (-k-1)z + z^2}$$

$\begin{matrix} \uparrow & \uparrow & \uparrow \\ c & b & a \end{matrix}$

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$\frac{-(-k-1) \pm \sqrt{(-k-1)^2 - 4 \cdot 1 \cdot k}}{2 \cdot 1} = 0$$

$$\frac{k+1 \pm \sqrt{k^2 + 2k - 1 - 4k}}{2} = 0$$

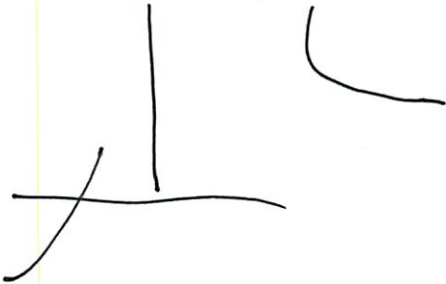
(4)

$$\frac{k+1 \pm \sqrt{k^2-2k-1}}{2} = 0$$

+ / No solutions exist



- / $k = -\frac{1}{2}$



~~real and negative & alternate signs~~
~~regain < 1 & bounded stable~~

This value of k causes ^{the} dominate pole to be 0,
causing the function to converge to 0

Test in Python

Try all the forms we knew

I get dominate pole ($k=1$) $\rightarrow 0$

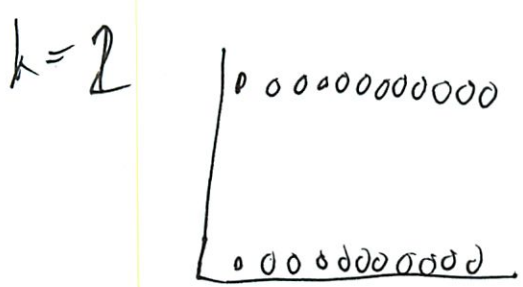
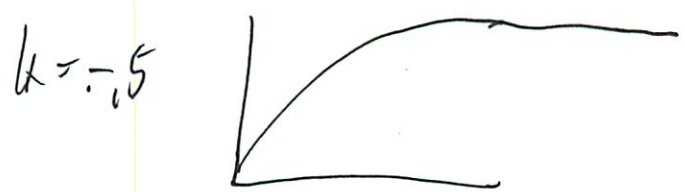
- why?

- am I doing model wrong?

best point $k=1$
scpt = $1.98 \cdot 10^{-15} = 0$

Got sm working w/ $init = 0$

9



What if I messed up controller system

Instead



Best $k = .4$

dominate pole = .63

What was those practice tests?

$$T[t+1] = T[t] + \Delta[t]$$

$$\Delta[t] = T_0 - T[t-1]$$

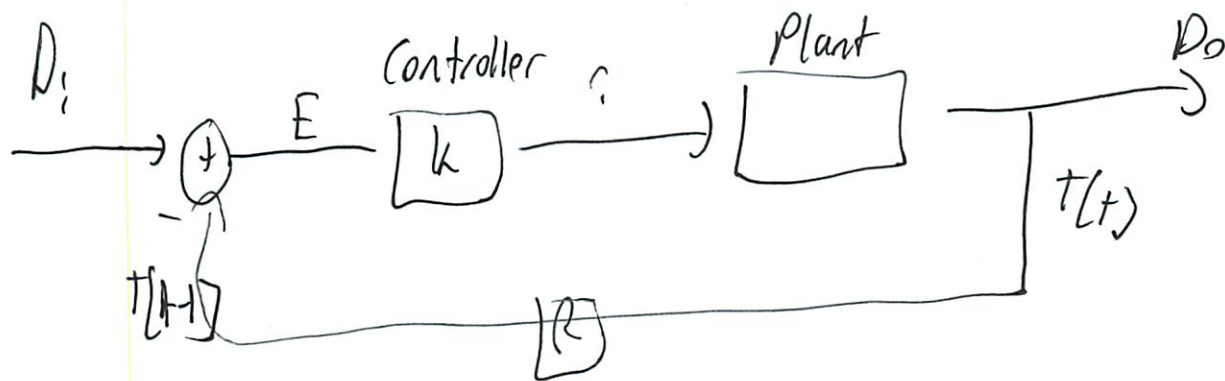
$$T[t+1] = T[t] + T_0 - T[t-1]$$

$$T[t] = T[t-1] + T_0 - T[t-2]$$

$$T[1-R-R^2] = T_0$$

What is the input + output of system

6



$$E = D_i - T[t-1]$$

What does Controller output?

- Robot velocity
angular velocity
- * Burner intensity or vent
Folk that makes more sense now

So plant

Intensity $\xrightarrow{\quad ? \quad} T[t]$

$T[n-1] + I \cdot T[n-1]$
 \uparrow since changed since now? \leftarrow also a delay from lecture notes

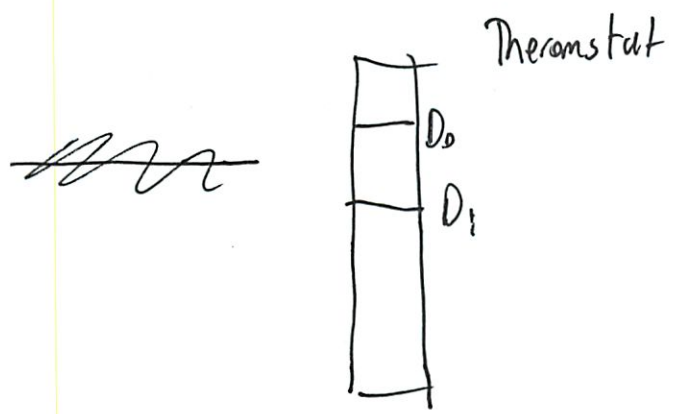
$$T[t] = T[t-1] + I[t-1]$$

$$T(1-R) = + D_o - t[n-1]$$

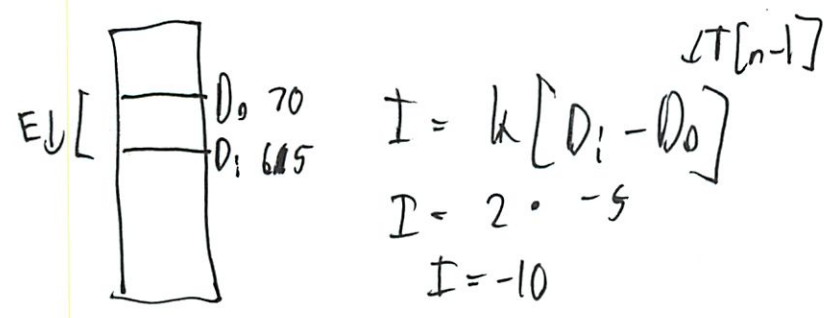
$$\begin{aligned}
 I &= D_o - t[n] \\
 I[n-1] &= D_o - t[n-1]
 \end{aligned}$$

② $T(1-2R) = D_0$

$\frac{T}{D_0} = \frac{1}{1-2R}$



Why can't I do this



Want to get to Desired
~~drop temp 10 degrees~~
 open vent 10 notches

$I = -10$ $T[n] + I$ $\xrightarrow{65}$
 $T[n] = 70$

$T[n+1] = T[n] + I$ ← just plant $\frac{I}{R}$ $\rightarrow T[n]$
 $T[n+1] = T[n] - k[D_i - T[n-1]]$ ~~error~~
 $T[n] = T[n-1] - k[D_i - T[n-2]]$ ~~if add R~~

← on so this is whole top already
 desired never changes - in lab they do

⑧ top
 D_0
 $\frac{D_0}{D_i}$
 Error
 is input
 here

whole system

$$\frac{D_0}{D_i}$$

from robot example this is $D_i R$
 - but desired is always the same?

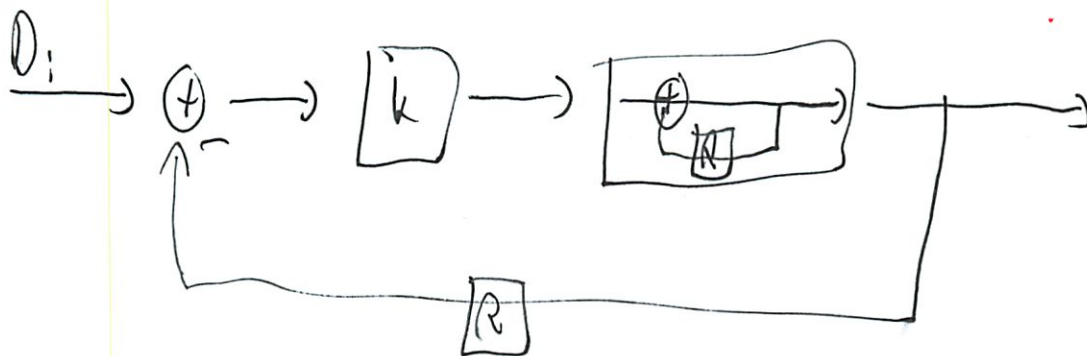
$$\rightarrow D_0 = D_0 R - k [D_i R - R^2 D_0]$$

\downarrow forget

$$D_0 = D_0 R - k D_i + k D_0 R^2$$

$$D_0 (1 - R - k R^2) = -k D_i R$$

$$\frac{D_0}{D_i} = \frac{-k R}{-k R^2 - R + 1}$$



- Same as I had before in diagram + code
- different function
 - I had probably did something wrong
 - used a different way

Current pos = $D[n] - D[n-1]$
 - but lets not care on this
 well I see where could
 get R when you
 delay everything
 des lab 5 does it
 - but it should not
 change pole so
 who cares

(9)

So now solve on paper

$$R = \frac{1}{z^2}$$

$$\frac{-k \left(\frac{1}{z^2} \right)}{-k \left(\frac{1}{z^2} \right) - \left(\frac{1}{z} \right) + 1}$$
$$\frac{-k}{-1 - z + kz^2}$$

Red not right either

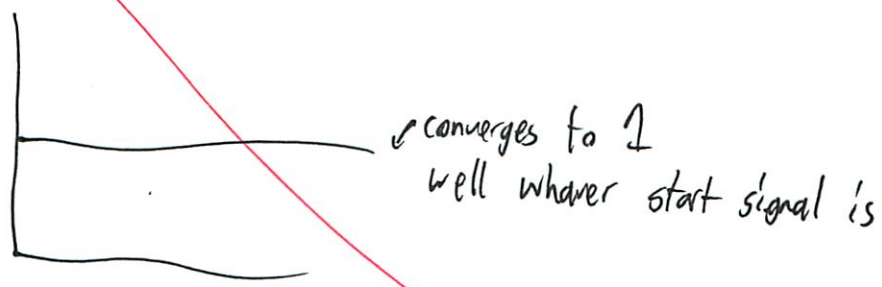
$$a = k$$
$$b = -1$$
$$c = -1$$

$$x^2 - x - 1$$
$$(x+1)(x-1)$$

$$\text{roots} = 1, -1$$

this matches best $k = 1$
dominate pole = 0

Plot gives



That says it never adjusts the temp, right?

But what I have matches the robot for plant

10

but should k go in denom then

$$\frac{k(R)}{kR^2 - R - 1}$$

← new
↓

and R on k
- due to sensor delay

I see where I messed up

$$a = k$$

$$b = -1$$

$$c = -1$$

$$\frac{1 \pm \sqrt{1 - 4 \cdot k - 1}}{2k}$$

$$\frac{1 \pm \sqrt{5}}{2k} = 0$$

+ / false

- / false

Unless does the R go on top in plant

$$\text{Since } D: [n-1] - T[n-2]$$



In that case $k = .4$ at pole .623



(11)

Should go ask in OH

But then that should not affect my SF

So the code can give out SF

$$\frac{4R}{1}$$

had bad - signs before?

$$4R^2 - 1R + 1$$

let me try this

$$4\left(\frac{1}{2}z\right) \cdot z^2$$

$$4\left(\frac{1}{2}z\right)^2 - 1\left(\frac{1}{2}z\right) + 1 \cdot z^2$$

$$\frac{4}{1} - z + z^2$$

$$a = 1$$

$$b = -1$$

$$c = k \text{ or oh c gets k right + the flip}$$

And where did 2 minus signs go

- diff sign for z

I think it was I messed up what got z
- did that before

(12)

one last try

$$\frac{1 \pm \sqrt{1 - 4 \cdot 1 \cdot k}}{2}$$

$$\frac{1 \pm \sqrt{1 - 4k}}{2} = 0$$

still false, 0

WTF?

will ask in Otl

and if $k = .4$ non real

Under my original diagram

$$\frac{k}{-1.5kR + 1}$$

so that clearly wrong

But why can't I solve this one on paper!

$$y[n] = y[n-1] - .4 y[n-2] + .4 x[n-1]$$

Yeah what I got

? need to $n-1$

If do dominate pole on $k = .4$ get non real result

Oh wait now it is getting $k = .26$
pole = .5049 what changed?
I should get that

(13)

or $k = .25$
pole = .5

$$\frac{k}{kR^2 - R + 1}$$

swap

$$1z^2 - R_2 + k$$

$$1 = a$$

$$-1 = b$$

$$k = c$$

$$\frac{1 \pm \sqrt{1 - 4 \cdot 1 \cdot k}}{2} = 0$$

$$\frac{1 \pm \sqrt{1 - 4k}}{\pm 1} = 0$$

$$\frac{\sqrt{1 - 4k}}{2} = 0$$

$$1 - 4k = 0$$

$$-4k = -1$$

$$k = \frac{1}{4} = .25 \text{ \textasciitilde Here we go}$$

why did I not get this before - had calc solve it
not me

(184)

So review issues

- did not delay D_i desired
Even though need to despite it not changing w/ time
- making ~~math~~ math mistakes converting to z
- actually did fairly good figuring out scenario
 - did bad at 1st math
 - w/ nom/denom
- bad at that 1st cascade thing
 - I think I did not break plant down all the way
 - actually it was almost correct
 - how could I have done that right
- do SF for each part

forgot blocks is $1 \oplus$

Controller

$$\frac{k}{1}$$

plant

$$\frac{\textcircled{R}}{-R+1}$$

Sensor

$$\frac{R}{1}$$

So

controller plant

$$= \frac{k}{1} \cdot \frac{R}{-R+1} = \frac{k \textcircled{R}}{-R+1}$$

did not have before

(15)

Then for Feedback subtract

$$\frac{\cancel{kR}}{-R+1}$$

$$1 - \frac{k}{1} \cdot \frac{\cancel{kR}}{-R+1} = 1 - \frac{kR}{1-R}$$

$$\frac{\cancel{kR}}{-R+1} \cdot \left(\frac{1}{1} - \frac{\cancel{kR}}{kR} \right)$$

$$\frac{kR^2}{(-R+1)(1-kR)}$$

$$\frac{kR^2}{-R - kR^2 + 1 - kR}$$

$$\frac{kR^2}{-kR^2 - kR - R + 1}$$

(same as I had!)

So what went wrong

the R^2

but now what

does not matter

Something wrong here
Wolfram considers the entire thing denom
 $(1-R)(1 - \frac{kR}{1-R})$

So what is that

$$1 - R - \frac{kR}{1-R} - \frac{kR^2}{\cancel{R-R^2} \text{ just denom } 1-R}$$

$$\frac{-kR^2 - kR}{1-R} + 1 - R$$

$$\frac{-kR^2 - kR}{1-R} + \frac{1-R}{1-R}$$

$$\frac{-kR^2 - kR + 1 - R}{1-R}$$

Oh is done

- but how does it add up

Oh its $1 +$
black's formula

(16) Try 1 +

$$\frac{\frac{kR}{-R+1}}{1 + \frac{kR}{1-R}}$$

$$\frac{kR}{1-R} \cdot \left(1 + \frac{1-R}{kR} \right)$$

$$\frac{kR + kR - kR^2}{1-R + kR - kR^2}$$

still not it

Oh I mis wrote the sensor as $\frac{k}{1}$ not $\frac{R}{1}$

$$\frac{\frac{kR}{1-R}}{1 + \frac{R}{1} \cdot \frac{kR}{1-R}}$$

$$\frac{kR}{1-R} \cdot \left(\frac{1}{1} + \frac{R}{R} \cdot \frac{1-R}{kR} \right)$$

$$\frac{kR + kR - kR^2}{1-R + kR^2 - kR^3}$$

(17) Now factor R at

$$\frac{R(2R - kR)}{1 - R}$$

Can't

Can you cancel any?

Actually no R up top

- but needs to be

- never mind works out

Should work

$$\frac{\frac{kR}{1-R}}{1 + \frac{kR^2}{1-R}}$$

← Wolfram alpha ✓

$$\frac{kR}{1-R} \cdot \left(1 + \frac{1-R}{kR^2}\right)$$

$$\frac{kR + kR - kR^2}{1-R + kR^2 - kR^3}$$

← wolfram alpha ✗

how about simplifying like this

$$\frac{\frac{kR}{1-R}}{\frac{1-R + kR^2}{1-R}}$$

$$\rightarrow \frac{kR}{\cancel{1-R}} \cdot \frac{\cancel{1-R}}{\cancel{kR^2} - R + 1}$$

= Here we go
I was just simplifying
wrong

⑧ On the very last page just forgot $\Delta[n-1]$
- but why is it like that again

$$T[n+1] = T[n] + \Delta[n]$$

$$T[n] = T[n-1] + \Delta[n-1]$$

? *

$$\Delta[n] = D_i - T[n]$$

$$\Delta[n-1] = D_i[n-1] - T[n-1]$$

~~***~~

and learn algebra!

will write clean copy now

Oh you can make a SM from a SF - cool

Part 2 Human in the loop

- oh its just simple delay
- like der lab 6

? but I am confused what it means that
whob is an accumulator

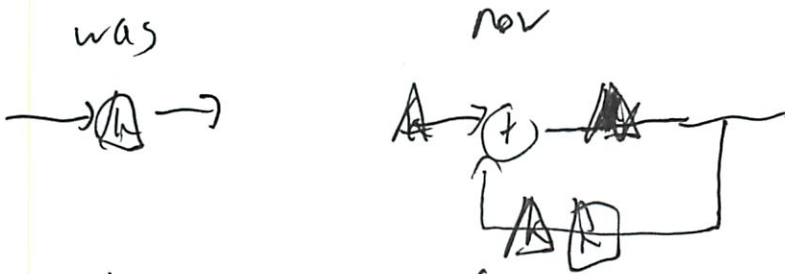
is controller just $B[n] = k_1 E[n] + k_2 E[n-1]$

this fits pattern we learned, ? human
but does it match scenario lagging

19

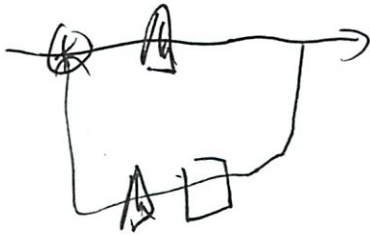
plant + sensor same

controller
was



is that deslab 6

had



do we multiply

k_1 in there again

Feedforward Add(k_1 , Cascade(k_2 , R))

Look at formula

how does help

prob same reason R was on top in last one

So

Controller

$$Y = k_1 x + k_2 x R$$

$$\frac{Y}{X} = \frac{k_1 + k_2 R}{1}$$

(20)

plant same

$$\frac{R}{-1R+1}$$

Sensor same

$$\frac{R}{1}$$

Cascade controller + plant

$$\frac{(k_1 + k_2 R)R}{-R + 1}$$

Feedback subtract

$$\frac{k_1 R + k_2 R^2}{1-R}$$
$$1 + \frac{k_1 R + k_2 R^2}{1-R} \cdot \frac{R}{1}$$

$$\frac{k_1 R + k_2 R^2}{1-R}$$
$$1 + \frac{k_1 R^2 + k_2 R^3}{1-R}$$

$$\frac{k_1 R + k_2 R^2}{1-R}$$

$$\frac{k_1 R + k_2 R^2}{\cancel{1-R}} \cdot \frac{\cancel{1-R}}{k_2 R^3 + k_1 R^2 - R + 1}$$

$$\frac{1-R}{1-R} + \frac{k_1 R^2 + k_2 R^3}{1-R}$$

(21)

$$\frac{k_1 R + k_2 R^2}{k_2 R^3 + k_1 R^2 - R + 1}$$

✓ perfect

? cant find poles by hand

- use computer

in Des Lab 6 they gave us k_1

k_1	k_2	prop	
10	-4.88	3.12	bad
1	-.38	.86	
.5	-.12	.12	better
.1	.099	.63	

Or is human changing T_i
Yeah

$$E = \underbrace{T_i[n]}_{\text{same}} + \cancel{\alpha_1} \alpha_2 [T_i[n] - T_0[n-1]]$$

$$\alpha_1 [T_0[n] - T_0[n-1]]$$

Oh basically same

but how T_0, T_i related - have always been same

2b

Cascade human system

$$\frac{k_2 R}{1-R} \cdot \frac{k_1 R}{k_1 R^2 - R + 1} = \frac{\cancel{R^2} k_1 k_2 R}{k_1 R^2 - R + 1 - k_1 R^3 + R^2 - R}$$
$$= \frac{k_1 k_2 R^2}{-k_1 R^3 + \underbrace{k_1 R^2 + R^2}_{k_2 R^2} - 2R + 1}$$

← according to running tutor - but why? just write for now

Feedback Subtract

running in P_Y

$$\frac{\cancel{R} k_1 k_2 R^2}{k_1^2 R^3 + k_2 R^2 - 2R + 1}$$

find roots

roots of denom

Can you find it if cubic

Flip

$$X^3 - 2X^2 + k_2 X + k_1^2 = 0$$

no quadratic formula

test values of k_1, k_2

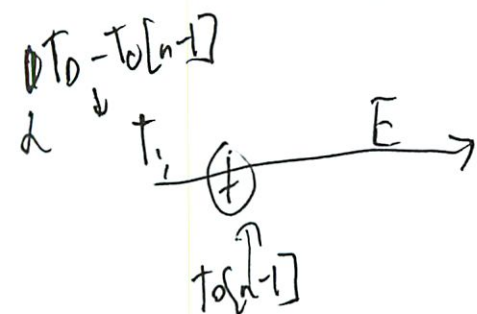
th

22

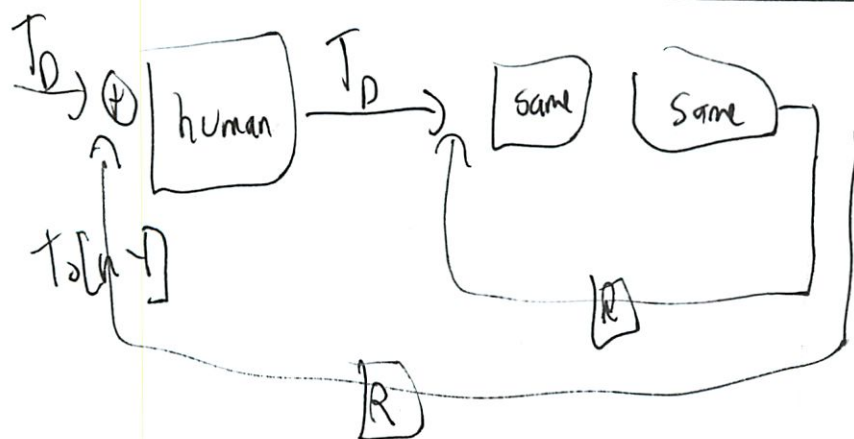
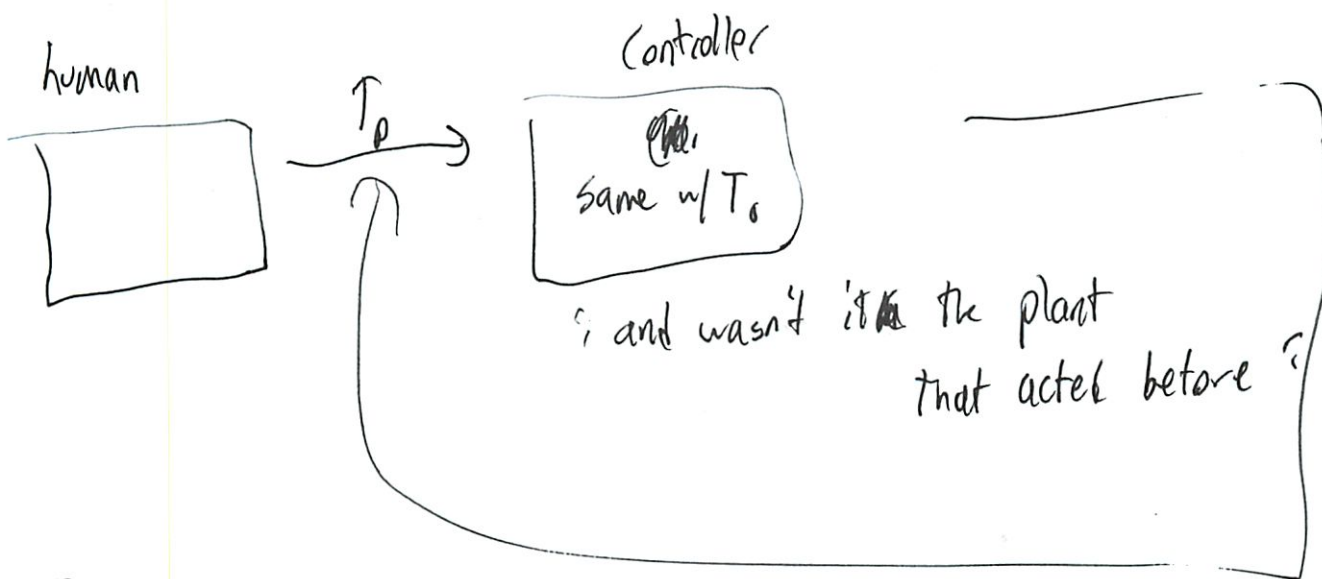
Or

$$E = \alpha_1 (T_0[n] - T_0[n-1]) + \alpha_2 (T_0[n] - T_0[n-1]) - T_0[n-1]$$

$\alpha_1 (T_i)$ $+ \alpha_2 (T_i - T_0) \leftarrow \text{last time}$



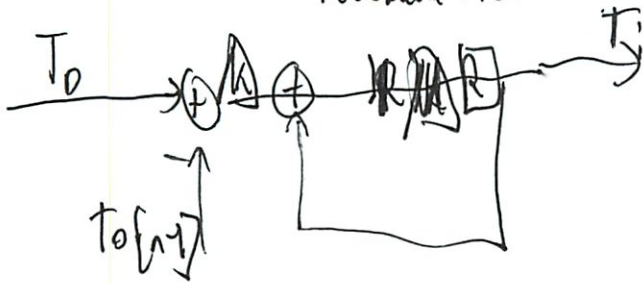
Or system thinking



(23)

human

Feedback Add?



$$T_i[n] = k(T_i[n] + T_0[n+1])$$

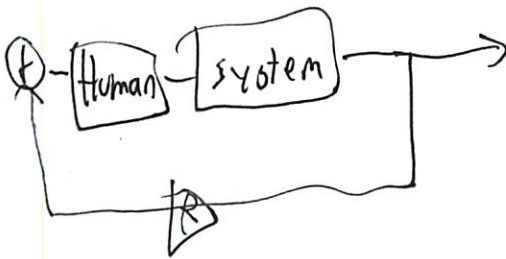
↑ accumulator

↑ would be $T_i[n] + T_i[n-1]$

$$= k(T_0 - T_0[n-1]) + T_i[n-1]$$

new adjustment old adjustments

New feedback subtract



cascade Human System

human

$$T_i[n] = k(T_0[n] - T_0[n-1]) + T_i[n-1]$$

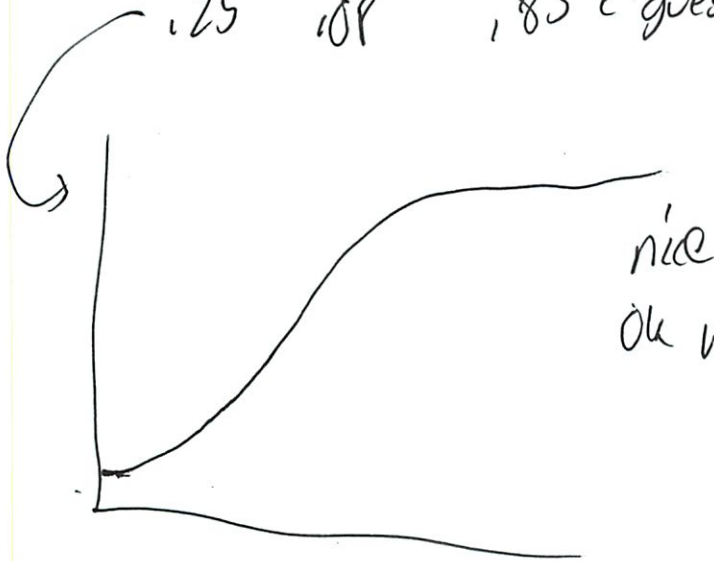
$$T_i(1-R) = k_2 E[n]$$

$$\frac{T_i}{E} = \frac{k_2 R}{1-R}$$

(2) 4

for	k_1	k_2	pole
	1	8	1
	3	0	6.73
	11	0.4	9.9
	0	1	-10

.25 .08 .83 guessing that is best



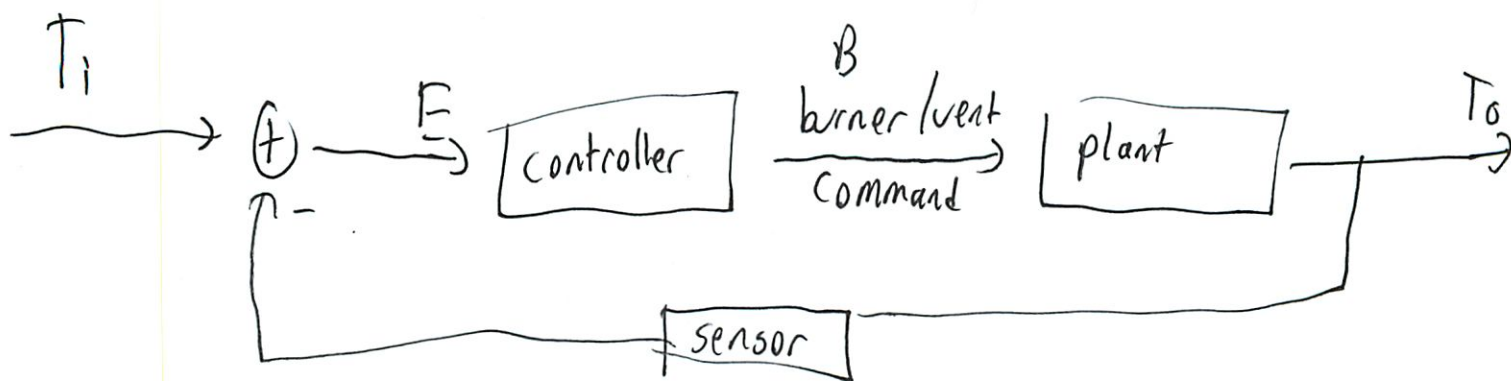
nice
Ok write it down

So was right on human?

~ key was separate

~~2/24~~

Oh their is an optimize over Grid function!



Controller

$$B[n] = k E[n]$$

$$E[n] = T_i[n] - T_o[n-1]$$

need time
for some
reason -
is this not
constant?

~~Does the controller
take a time step?~~

sensor

$$T_o[n] = T_o[n-1]$$

plant

$$T_o[n+1] = T_o[n] + E[n]$$

System

$$T_o[n+1] = T_o[n] + k(T_i[n] - T_o[n-1])$$

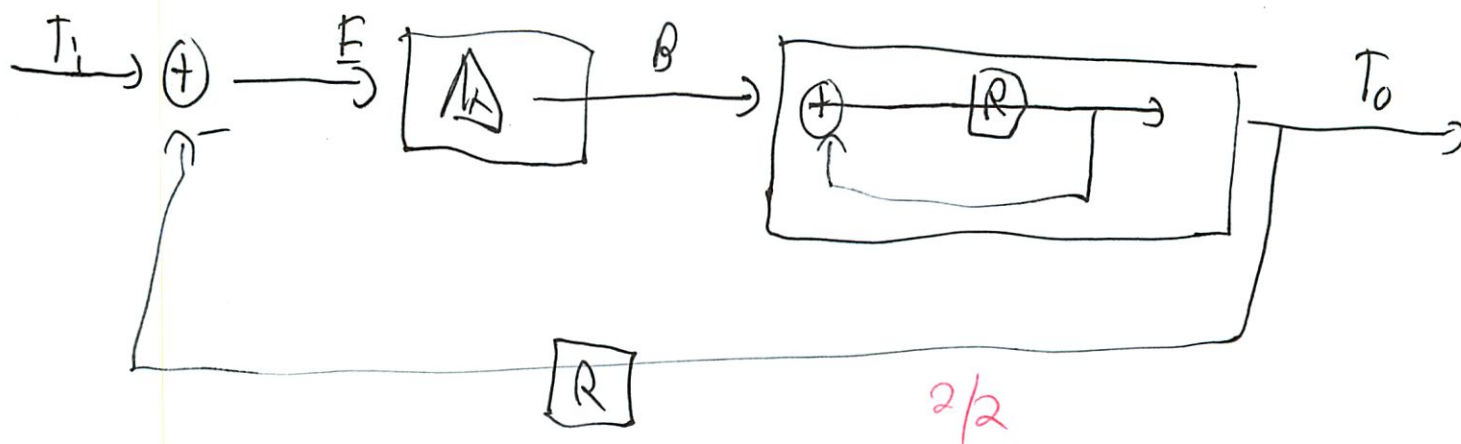
$$T_o[n] = T_o[n-1] + kT_i[n-1] - kT_o[n-2]$$

$$T_o R = T_o R + kT_i R - T_o k R^2$$

$$T_o(1 - R + kR^2) = kT_i R$$

$$\frac{T_o}{T_i} = \frac{kR}{kR^2 - R + 1} \quad 2/2$$

(2)



Best gain $\rightarrow R = \frac{1}{2}z^2$

$$\frac{kR}{kR^2 - R + 1}$$

$$\frac{k\left(\frac{1}{2}z^2\right)}{k\left(\frac{1}{2}z^2\right)^2 - \left(\frac{1}{2}z^2\right) + 1} \cdot z^2$$

$$\frac{k}{k - z + z^2}$$

$$a = 1$$

$$b = -1$$

$$c = k$$

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$\frac{-(-1) \pm \sqrt{(-1)^2 - 4(1)(k)}}{2(1)} = 0$$

$$1 \pm \sqrt{1 - 4k} = 0$$

3

$$\frac{\sqrt{1-4k}}{2} = 0$$

$$1-4k = 0$$

$$-4k = -1$$

$$k = \frac{1}{4} \text{ is the best gain}$$

$$\text{mag dom pole} = .5$$

Can also do it other way

Controller

$$\frac{B}{ME} = \frac{k}{1}$$

Plant

$$\frac{T_o}{B} = \frac{A}{-A+1}$$

Sensor

$$\frac{R}{1}$$

Cascade controller plant

$$H_1 H_2 = \frac{k}{1} \cdot \frac{A}{1-R} = \frac{kA}{1-R}$$

④

Feedback Subtract

$$\frac{H_1}{1 + H_1 H_2}$$

$$\frac{\frac{kR}{1-R}}{1 + \frac{kR}{1-R} \cdot \frac{R}{1}}$$

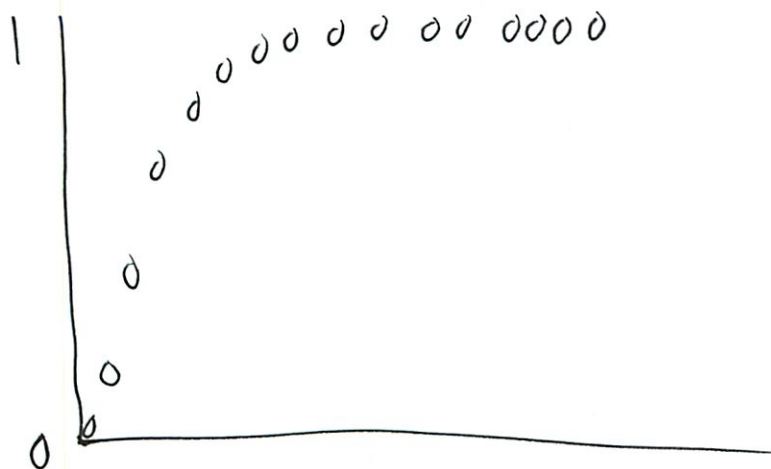
$$= \frac{\frac{kR}{1-R}}{\frac{1-R}{1-R} + \frac{kR^2}{1-R}}$$

$$= \frac{\frac{kR}{1-R}}{\frac{kR^2 - R + 1}{1-R}}$$

$$= \frac{\cancel{kR}}{\cancel{1-R}} \cdot \frac{\cancel{1-R}}{kR^2 - R + 1}$$

✓ Same as before

Plot

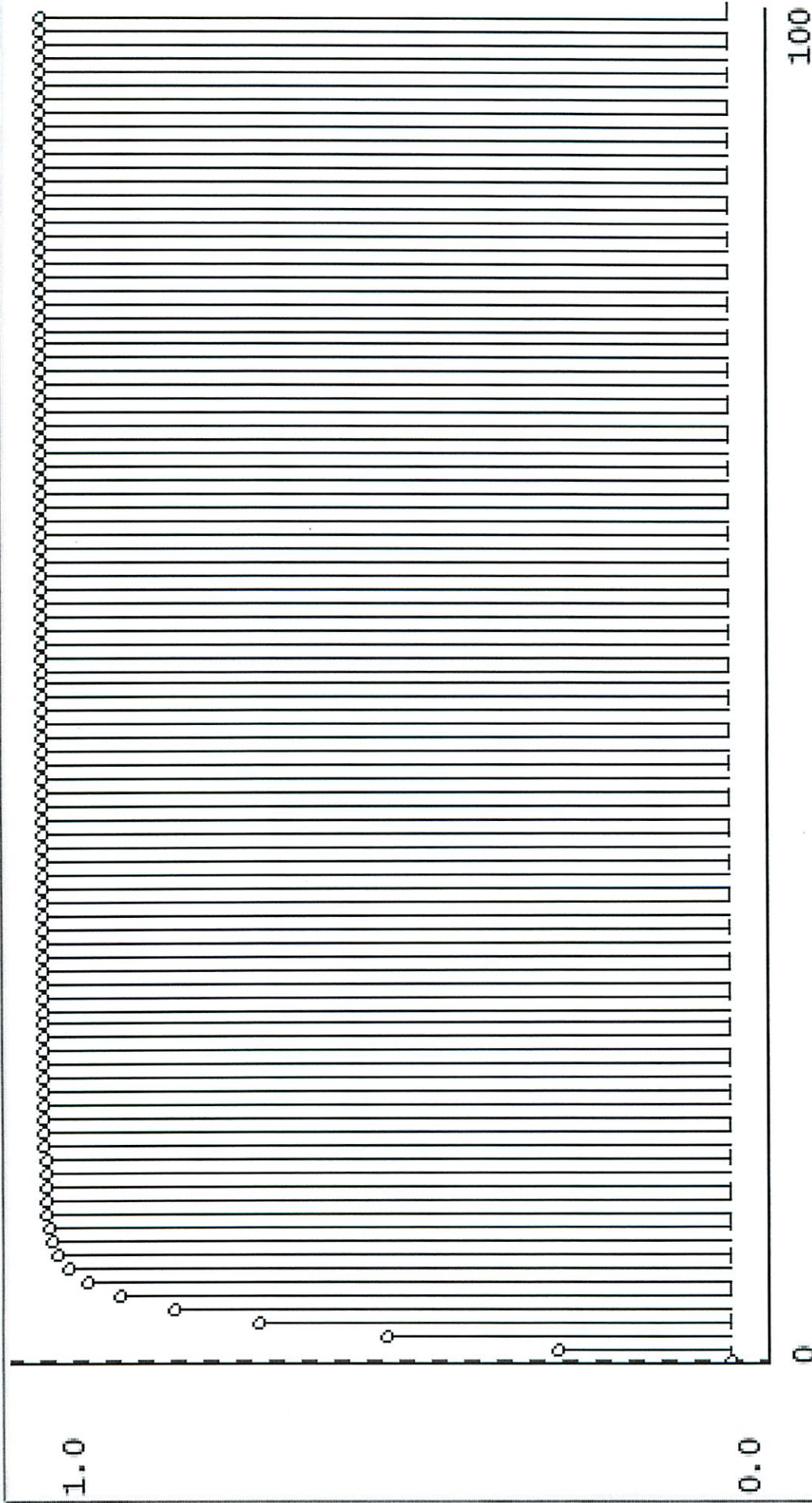


System
No Human

$k = 25$

Mag dom pole = 1.5

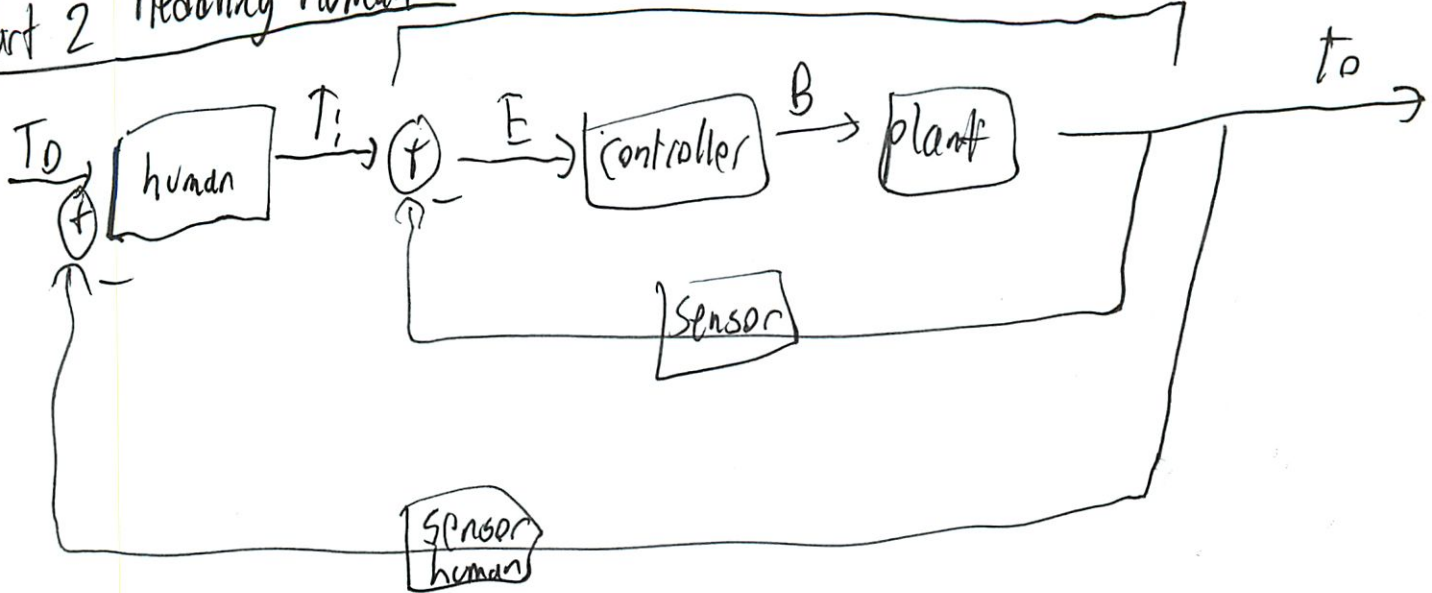
7% Signal value versus time (2010-10-17 11:25:44.172000)



5

Part 2 Modelling Human

Same



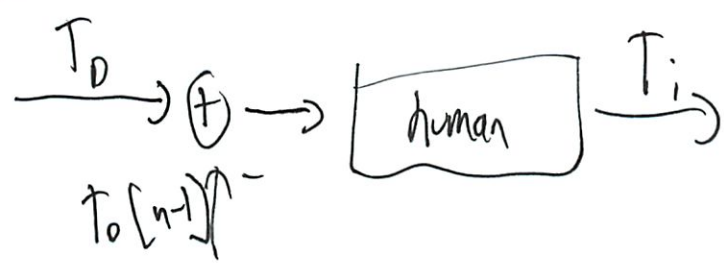
human sensor
Same as mechanical sensor

human sensor

Same as machine sensor

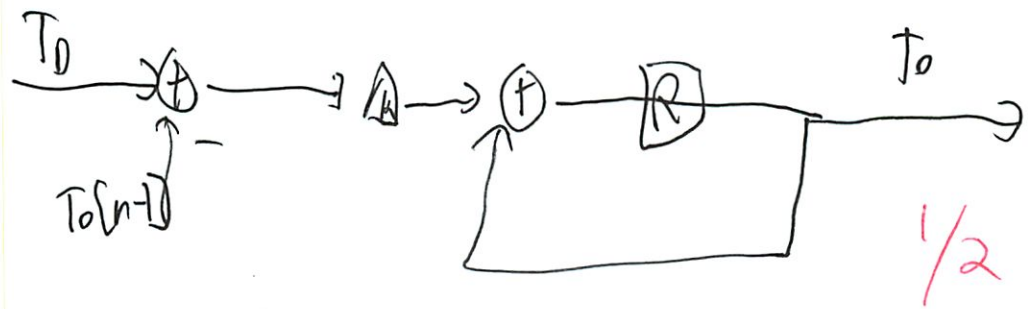
$$T_o[n+1] = T_o[n]$$
$$T_o[n] = T_o[n-1]$$

human

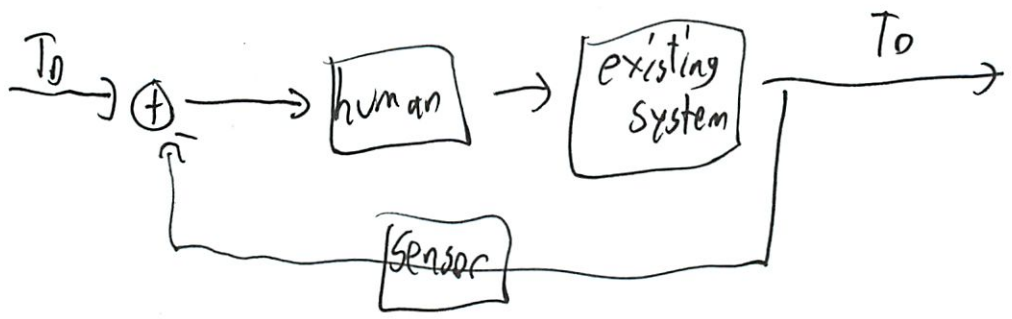


(6)

$$T_i[n] = \underbrace{k_2 [T_o[n] - T_o[n-1]]}_{\text{new adjustment}} + \underbrace{T_i[n-1]}_{\text{accumulator}}$$



So what we have is



~~Cascade Human & Existing system~~

Human

$$T_i = k_2 (T_o - T_o R) + T_i R$$

$$T_i (1 - R) = k_2 (T_o - T_o R)$$

$$\frac{T_i}{E} = \frac{k_2 \cancel{T_o} R}{1 - R}$$

⑧ Cascade Human + system

$$= \frac{k_2 R}{1-R} \circ \frac{k_1 R}{k_1 R^2 - R + 1}$$

$$= \frac{k_1 k_2 R^2}{-k_1 R^3 + k_2 R^2 - 2R + 1}$$

Feedback subtract

$$k_1 k_2 R^2$$

$$-k_1 R^3 + k_2 R^2 - 2R + 1$$

$$= \frac{1 + \frac{R}{1} \circ \frac{k_1 k_2 R}{-k_1 R^3 + k_2 R^2 - 2R + 1}}$$

$$= \frac{k_1 k_2 R^2}{k_1 R^3 + k_2 R^2 - 2R + 1}$$

1/2

Now need to find roots
- but can't since not quadratic
So use best k_2

Was not supposed to
do yet

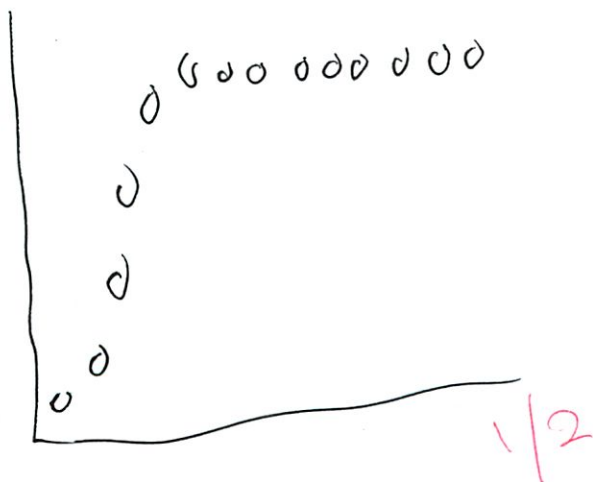
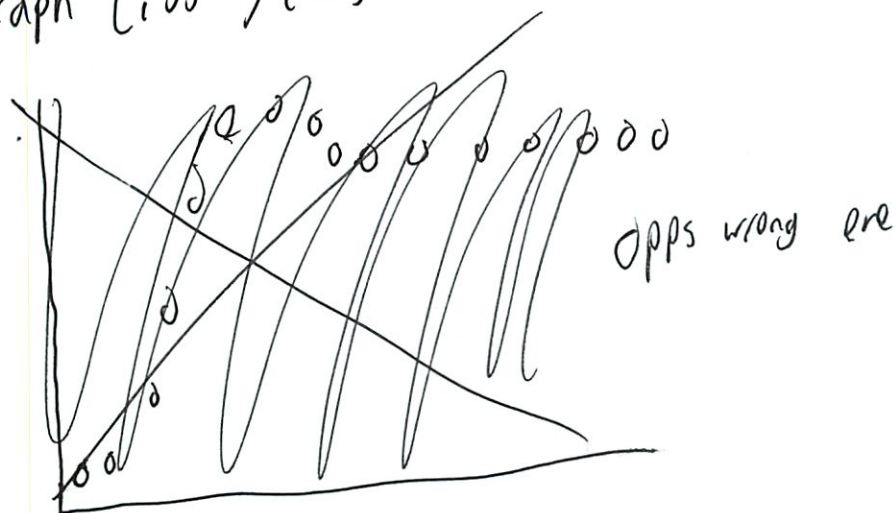
See Step 12

k_1	k_2	dom pole
0	1	18
.1	.04	.94
.25	.08	.83
.5	.15	.75

8

k_1	k_2	dom pole
.6	.16	.80
.4	.14	.71
.3	.1	.78
.35	.12	.70 ← best
.36	.14	.74
.34	.12	.72

Graph (.35, .12)



Dominate pole

$$.7 + 0j$$

$$.7$$

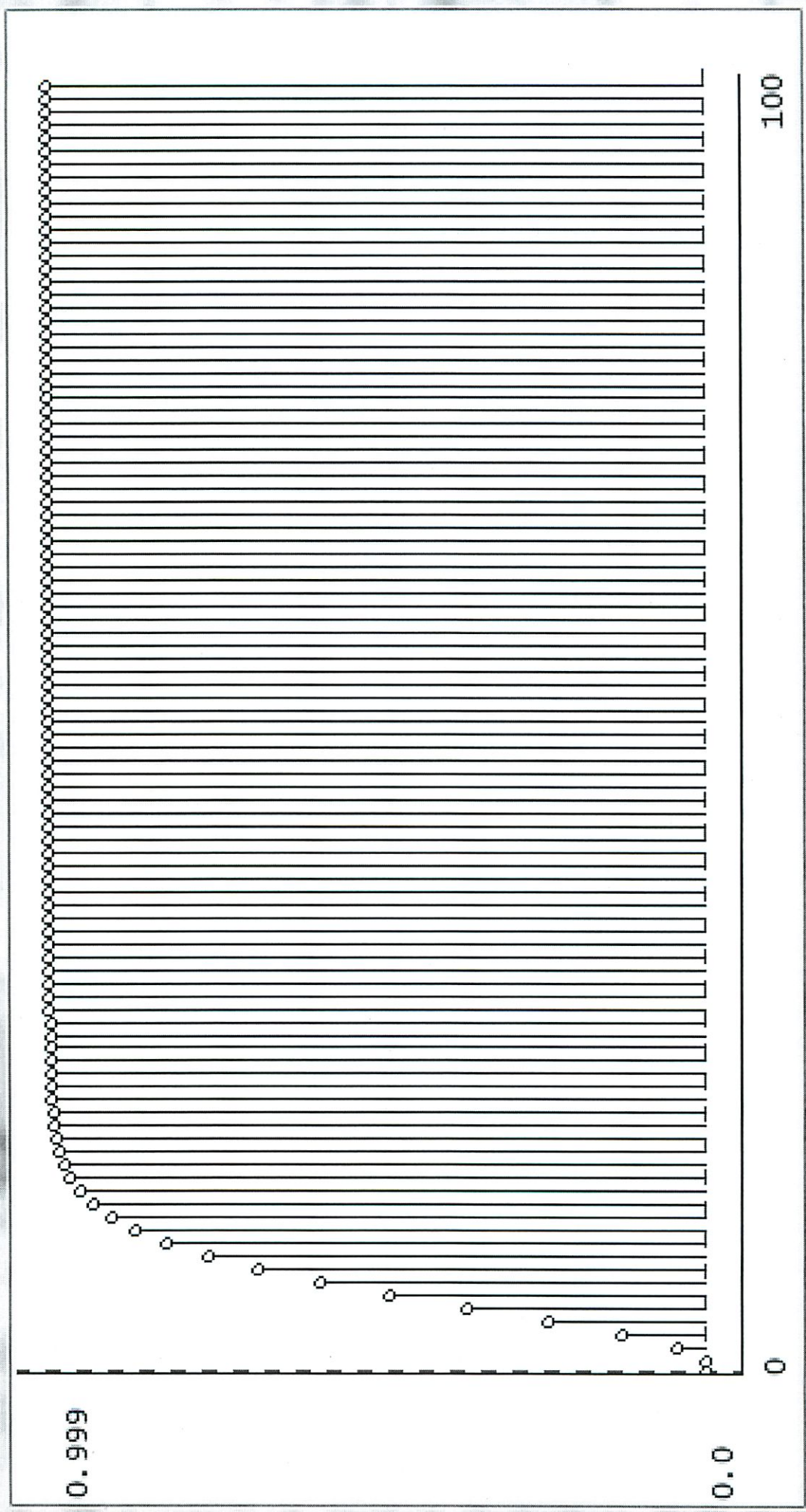
Human system

$$k_1 = .35$$

$$k_2 = .12$$

dominate pole = .70

7% Signal value versus time (2010-10-17 11:27:03.868000)



⑨ Just in case I forgot model for whole system

$$T_o[n] = 2T_o[n-1] - 1.35T_o[n-2] + 1.308T_o[n-3] + 1.042T_o[n-2]$$

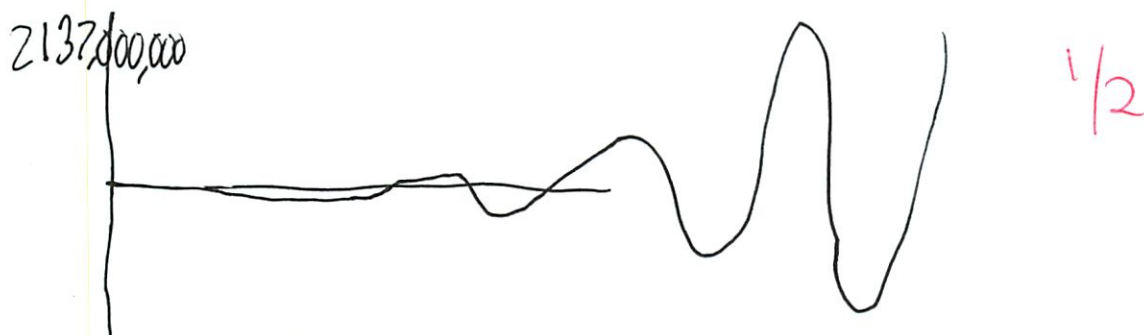
or in generic terms

$$T_o[n] = 2T_o[n-1] - k_2T_o[n-2] - (k_1)^2T_o[n-3] + (k_2)^2T_o[n-2]$$

Step 8

So if we have

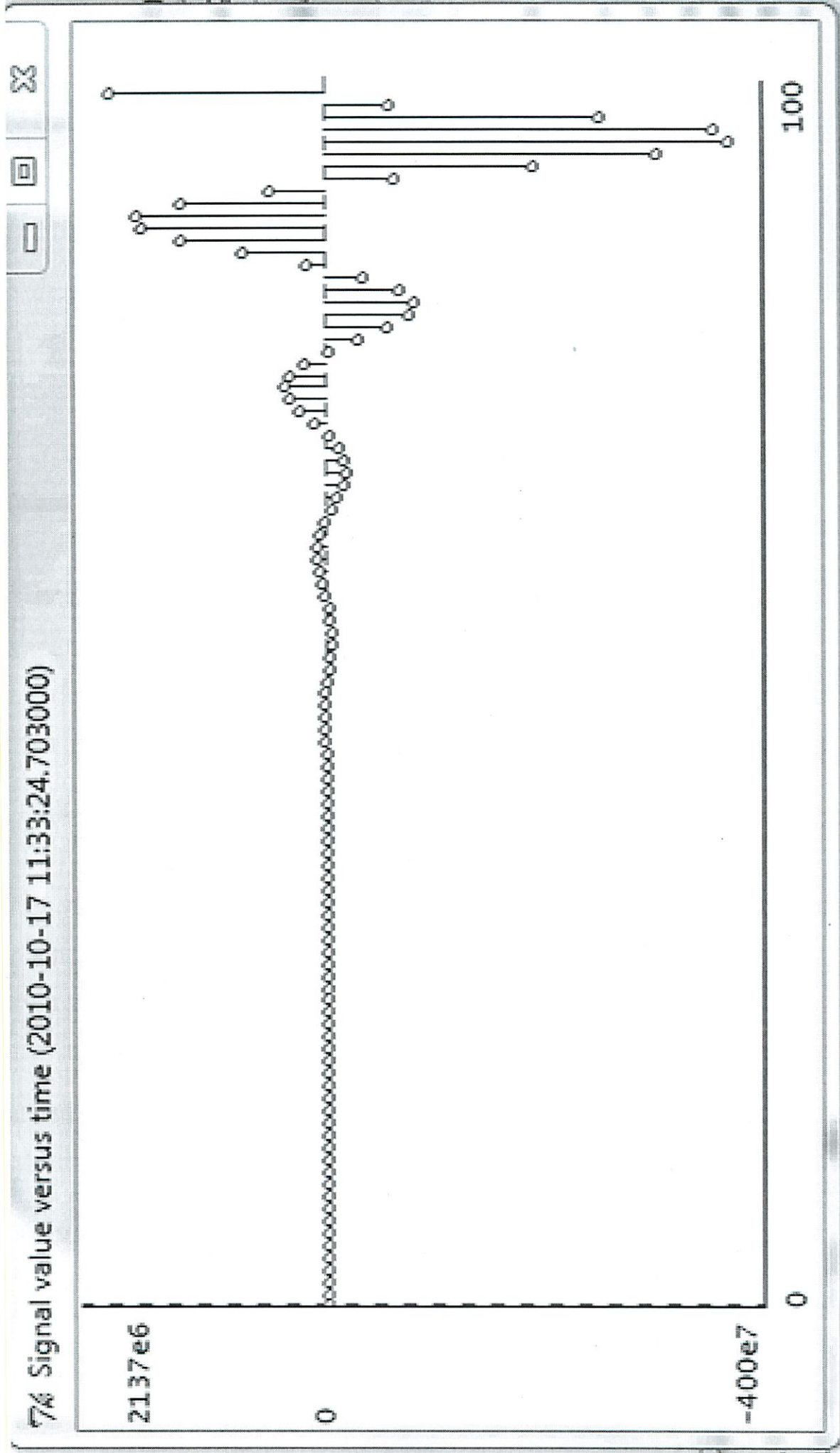
~~Then~~ $k_1 = .25 \quad k_2 = 1$



unbounded anyway
very bad idea

The human would be setting the thermostat ~~too~~ by too big intervals causing the furnace to miss and overshoot. They could never correct and the temperature oscillates out of control.

Human Step 8 $k_1 = 25$
 $k_2 = 1$

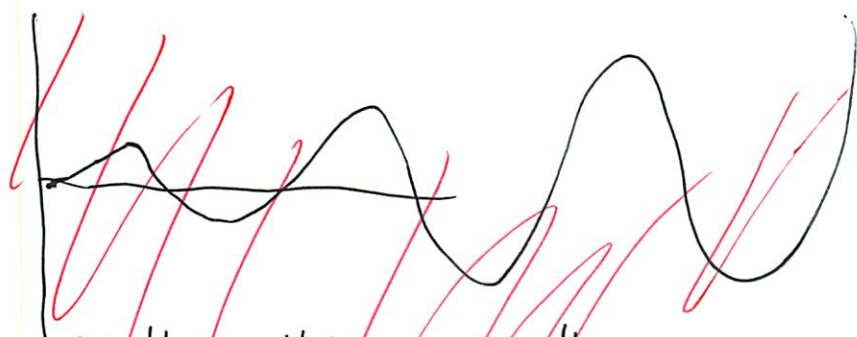


(10) Step 4

Change best k_2 to best k_1 and run best k_1 at $k_2=1$

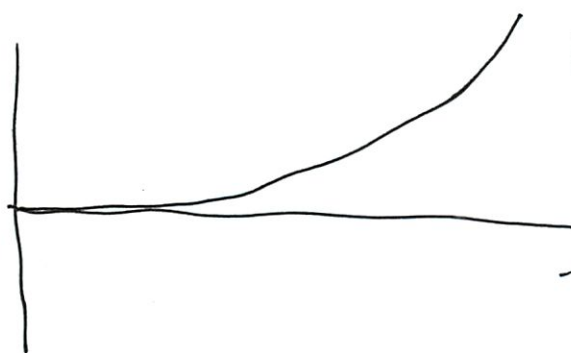
best ~~k_1~~ $k_1 = 8.2 e^{-15}$

dominate pole = 1.0000004



Would oscillate as well

Would not be possible



would increase

very slowly

almost no gain

- just let human control the burner directly, right?

$1/2$

But at pole of 1.0000004 the system would not act stably

- increasing monotonically, unbounded

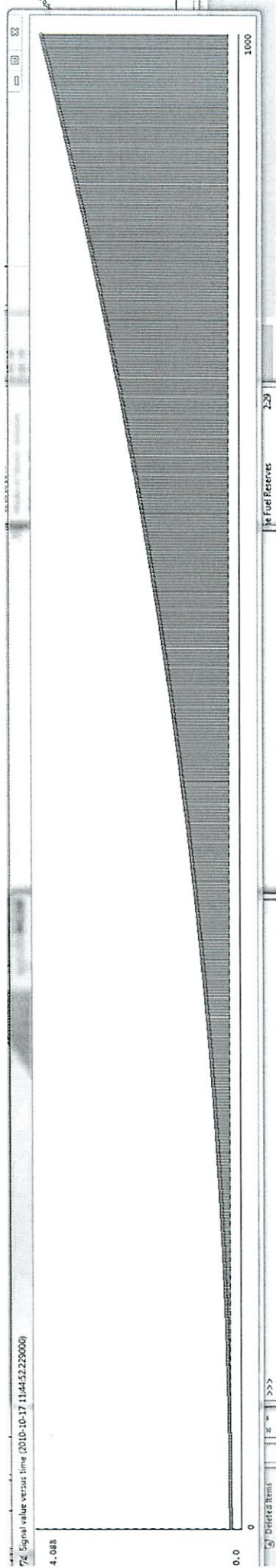
Wait when I ran it again I got a dominate pole of $.999999956$ - so it will just approach stability, taking forever

Human Step 9

$$k_1 = 8.2 \times 10^{-15} \approx 0 \text{ a bit more than}$$

$$k_2 = 1$$

$$\text{dominate pole} = 1,000,000 \approx 1$$



⑪:

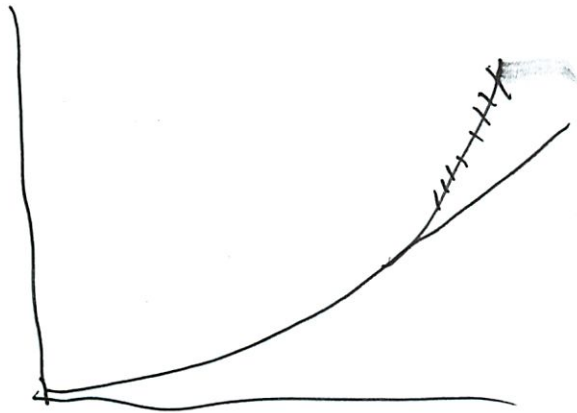
Step 10

best k_1 for k_2 of .5

$$k_1 = 8.20 e^{-15}$$

$$\text{mag dom pole} = 1$$

The same k_1 is recommended
That makes graph ~~about~~ the same



But this would take forever, making -people upset

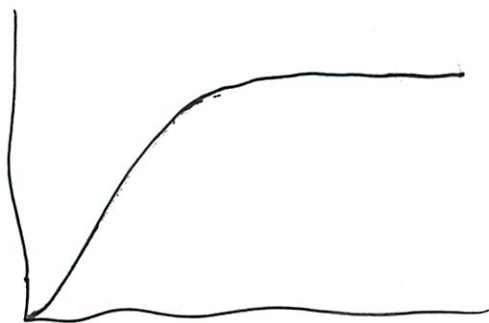
In both 11 and 10 → Furnace could not really operate
Stably in a way that
would make people happy

Step 11

best k_2 for k_1 of .25

$$k_2 = .076$$

$$\text{mag dom pole} = .8348$$



will be stable and
approach temp

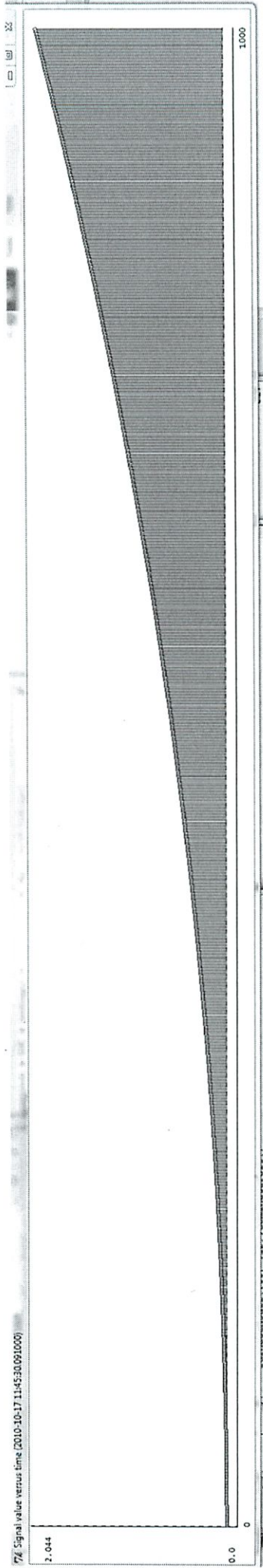
Human Step 10

↓ a bit more than

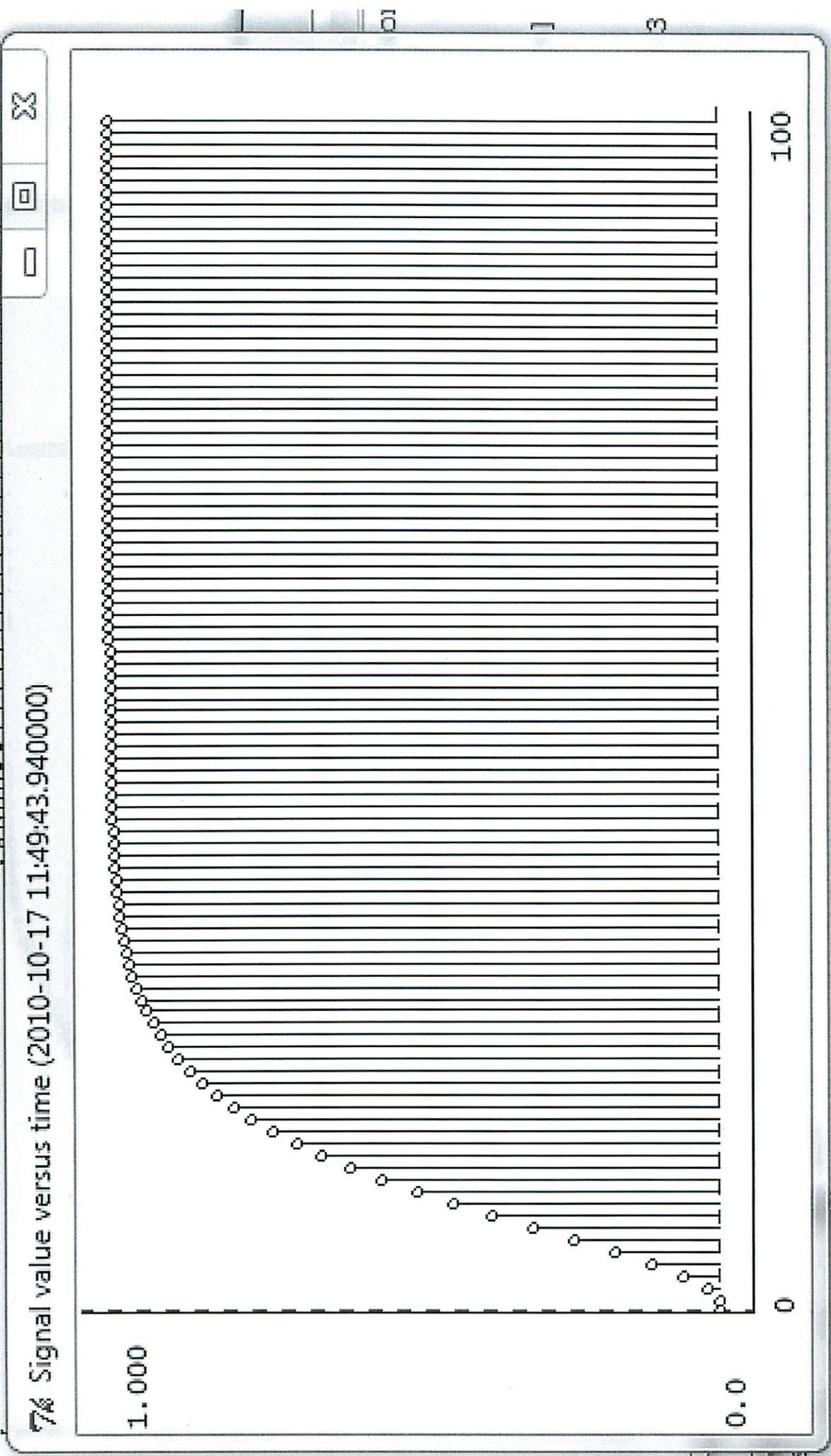
$$k_1 = 8.20 \times 10^{-15} \approx 0$$

$$k_2 = 1.5$$

$$\text{mag dom pole } 1.000004 \approx 1$$



Human Step 11 $k_1 = 0.25$ mag dam pole = ~~0.000~~ 8348
 $k_2 = 0.076$



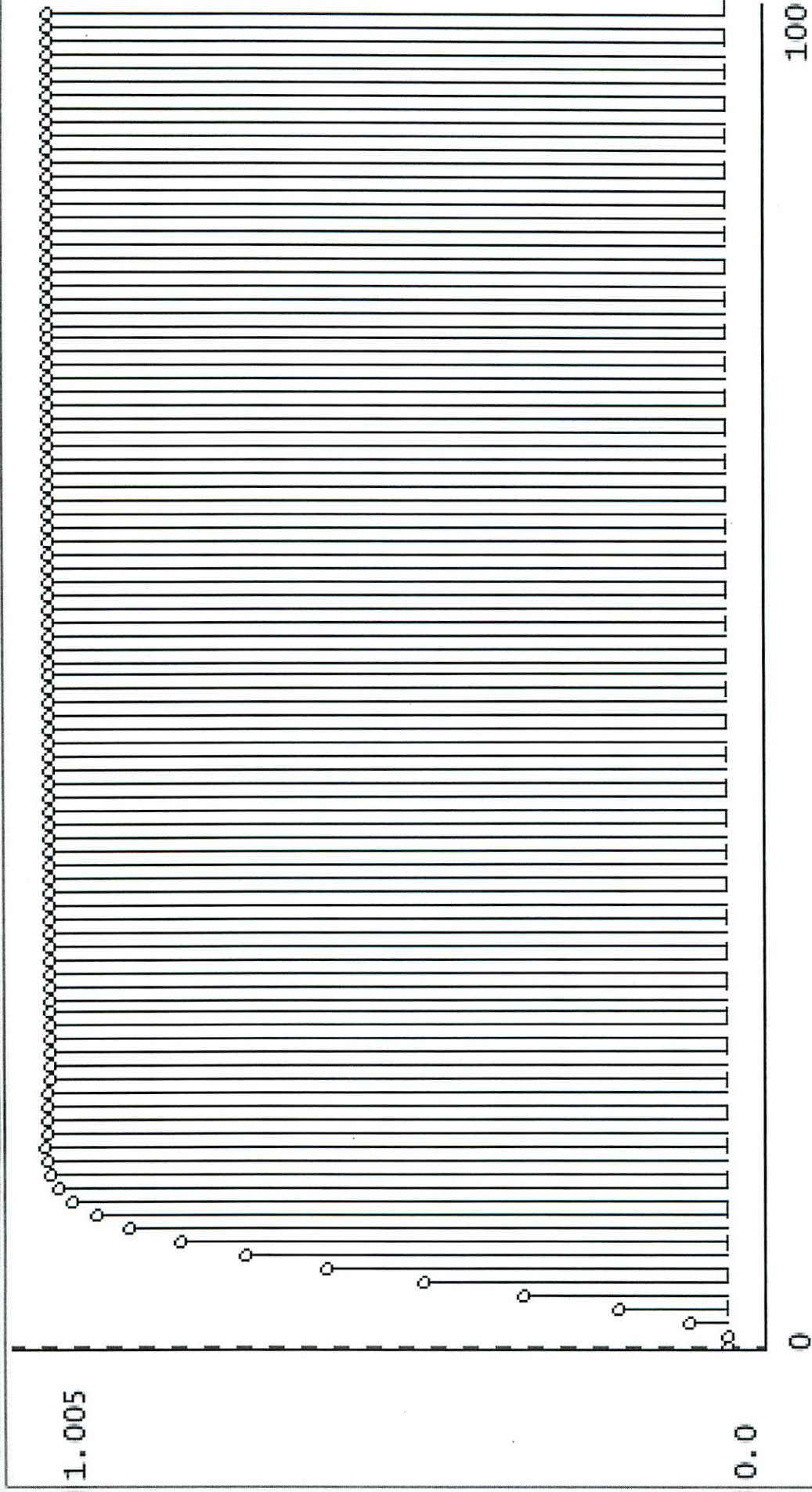
Human Step 12

$$k_1 = .38$$

$$k_2 = .14$$

mag dom pole ≈ 1.7

7% Signal value versus time (2010-10-17 11:57:00.678000)



⑫ Step 12

Best Paring w/ optimize over grid

~~OK~~ $k_1 = .38$

$$k_2 = .14$$

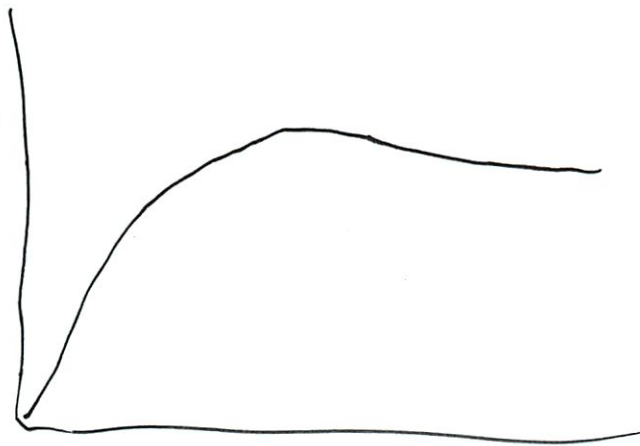
$$\text{max dom pole} = .70$$

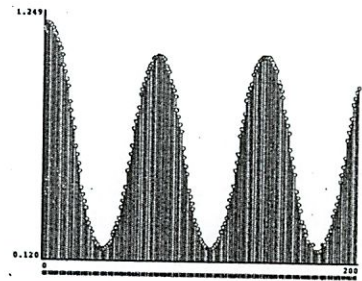
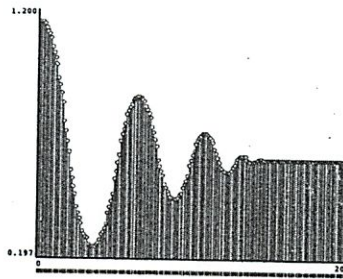
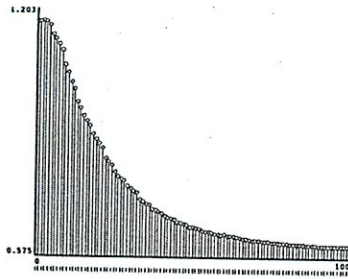
Better than just furnace alone?

No!, Human interfeears

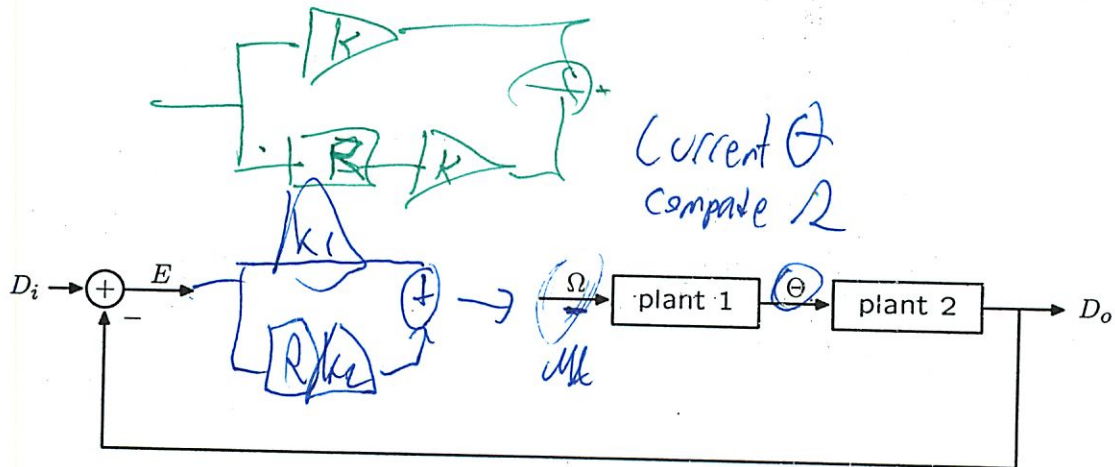
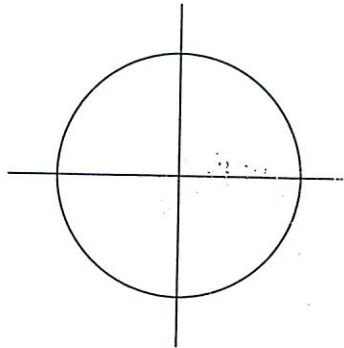
Plot .38, .14

1/2





bounded $|c|$
monotonic real
pos



$$\frac{r}{E} = \frac{k}{1}$$

$M_1 H_2$

6.01: Introduction to EECS I

Circuits

Q3

Week 7

October 19, 2010

PCAP - Same Themes are back

What is a circuit?

A path that starts at some location, and after traversing other parts of the system, returns to that location.

Typically, traversal of the circuit involves transmission of some entity:

- vehicles
- electrical "force" *charge/current*
- blood or other fluids
- information

closed loop

Circuits: A Very Different PCAP System

Circuits provide us with a very different kinds of primitives, composition, abstraction, and patterns.

*Some differences
flows*

Constraints, not input/output relationships

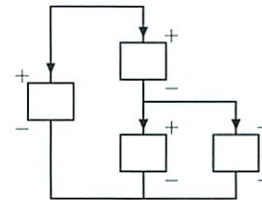
Reading: Sections 7.1 – 7.6.2

constraints?

The Circuit Abstraction

Circuits represent systems as connections of **components**:

- through which currents ('through' variables) flow and
- across which voltages ('across' variables) develop.

Reason about both*Still cases where building dedicated circuits
is worth the time*

The Circuit Abstraction

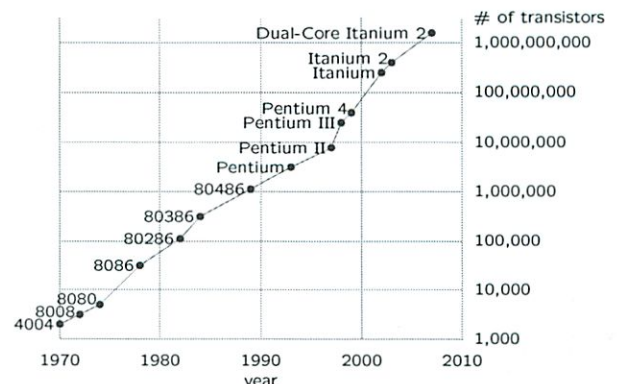
Circuits are important for two very different reasons:

- as physical systems
 - power (from generators and transformers to power lines)
 - electronics (from cell phones to computers)
- as models of complex systems
 - neurons
 - brain
 - cardiovascular system
 - hearing

Models of other complex systems

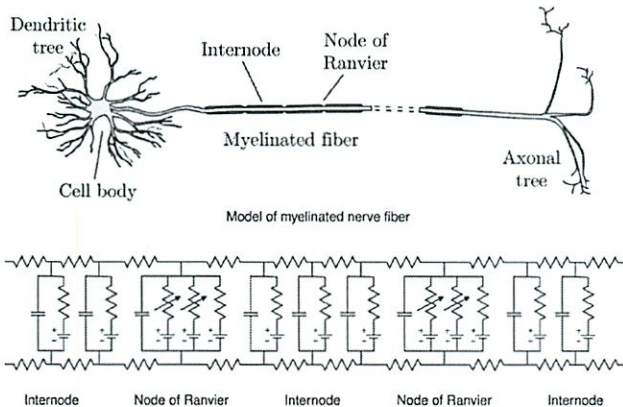
The Circuit Abstraction

Circuits are the basis of our enormously successful semiconductor industry.

*how would you design a processor w/o
abstracting details*

The Circuit Abstraction

Circuits as models of complex systems: myelinated neuron.



Modeling transition of information

Circuit Theory

Convenient to envision circuits as conducting fluid flow

- Compressible versus incompressible
- Air, vehicles (?) versus water, electricity

Characterize a circuit by:

- Conservation laws, governing currents and voltages in general
- Constituent equations, describing behavior of individual components

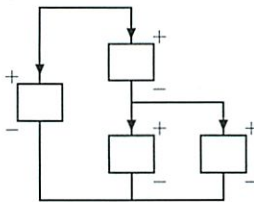
We will study only circuits with static behavior

push in one place → out other end

What is a Circuit?

Circuits are connections of components

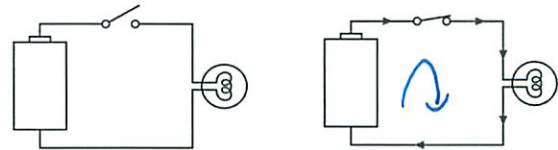
- through which currents ('through' variables) flow and
- across which voltages ('across' variables) develop.



Rules Governing Flow

Rule 1: Currents flow in loops.

Example: flow of electrical current through a flashlight



When the switch is closed, electrical current flows through the loop.

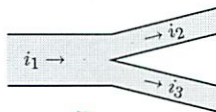
The same amount of current flows into the bulb (top path) and out of the bulb (bottom path).

current preserved around loop

Rules Governing Flow

Rule 2: Like the flow of water, the flow of electrical current (charged particles) is incompressible.

Example: flow of water through a branching point



What comes in must go out.

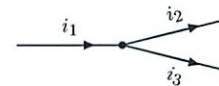
No deep well

Here $i_1 = i_2 + i_3$.

Kirchoff's Current Law: the sum of the currents into a node is zero.

Rules Governing Flow

In electrical circuits, we represent current flow by arrows on lines representing connections (wires).



$$i_1 = i_2 + i_3$$

The dot represents a "node" which represents a connection of two or more segments.

direction = signed variables



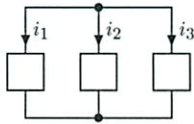
2

$$i_2 = i_1 - i_3$$

Nodes

Nodes are represented in circuit diagrams by lines that connect circuit components.

The following circuit has three components, each represented with a box.



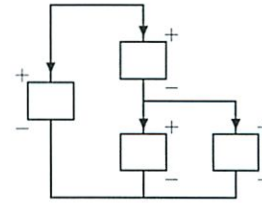
There are two nodes, each indicated by a dot. The net current into or out of each of these nodes is zero. Therefore $i_1 + i_2 + i_3 = 0$.

no current source here

What is a Circuit?

Circuits are connections of components

- through which currents ('through' variables) flow and
- across which voltages ('across' variables) develop.



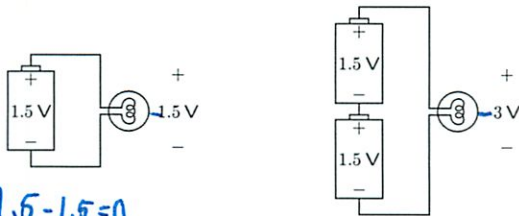
By analogy, voltages drive current flows. Difference in voltage is critical, absolute voltage is not

*flow of water b/w tanks
diff. in heights causes flow*

Rules Governing Voltages

Voltages accumulate in loops.

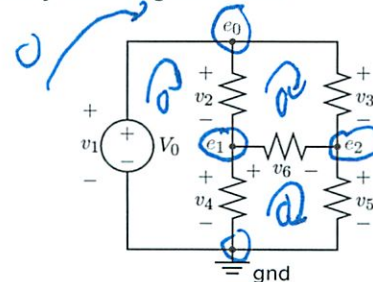
Example: the series combination of two 1.5 V batteries supplies 3 V.



Kirchoff's Voltage Law: the sum of the voltages around a closed loop is zero.

Alternative Representation: Node Voltages

Node voltages represent the voltage between each node in a circuit and an arbitrarily selected ground node.



Node voltages and component voltages are different but equivalent representations of voltage.

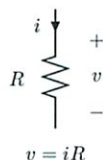
- **component voltages** represent the voltages across components.
- **node voltages** represent the voltages in a circuit.

all loops matter!

relative to fixed pt

Rules Governing Components: Resistor

Each component is represented by a relationship between the voltage across the component to the current through the component.

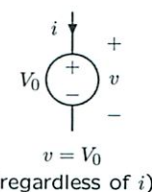


$$v = iR$$

same as 8.02

Rules Governing Components: Voltage source

Each component is represented by a relationship between the voltage across the component to the current through the component.



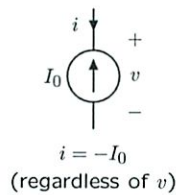
$$v = V_0$$

(regardless of i)

battery

** what want to use most***Rules Governing Components: Current source**

Each component is represented by a relationship between the voltage across the component to the current through the component.

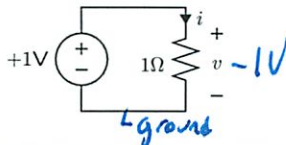
**Node-Voltage-and-Component-Current (NVCC) Method***(Kircoff's circuit rule)*

Combining KCL, node voltages, and component equations leads to the NVCC method for solving circuits:

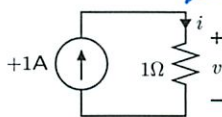
- Assign **node voltage variables** to every node except ground (whose voltage is arbitrarily taken as zero). *branch/connection*
- Assign **component current variables** to every component in the circuit.
- Write one **constitutive relation** for each component in terms of the component current variable and the component voltage, which is the difference between the node voltages at its terminals.
- Express **KCL** at each node except ground in terms of the component currents.
- **Solve** the resulting equations.

Analyzing Simple Circuits

Analyzing simple circuits is straightforward.



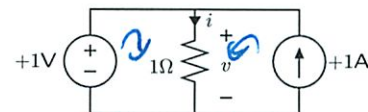
The voltage source determines the voltage across the resistor, $v = 1V$, so the current through the resistor is $i = v/R = 1/1 = 1A$.



The current source determines the current through the resistor, $i = 1A$, so the voltage across the resistor is $v = iR = 1 \times 1 = 1V$.

Check Yourself

What is the current through the resistor below?



1. 1A
2. 2A
3. 0A
4. cannot determine
5. none of the above

$$\frac{1V}{1\Omega} = 1A$$

So voltage source not providing any voltage

$F_1 \downarrow L_1 \quad L_2 \quad \downarrow F_2$ *output: anything that can be measured*

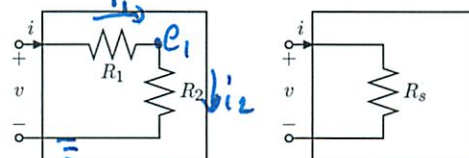
Common Patterns

There are a number of **common patterns** that facilitate design and analysis:

- series resistances
- parallel resistances
- voltage dividers
- current dividers

Series Combinations

The series combination of two resistors is equivalent to a single resistor whose resistance is the sum of the two original resistances.



$$v = R_1 i + R_2 i$$

$$v = R_s i$$

Voltage drop

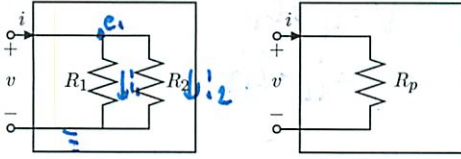
$$R_s = R_1 + R_2$$

The resistance of a series combination is always larger than either of the original resistances.

$$\begin{aligned} v - e_1 &= i_1 R_1 \rightarrow v = i_1 R_1 + i_2 R_2 \\ e_1 - 0 &= i_2 R_2 \\ i &= i_1 = i_2 \\ &= i(R_1 + R_2) \end{aligned}$$

*what's the equivalence***Parallel Combinations**

The parallel combination of two resistors is equivalent to a single resistor whose conductance (1/resistance) is the sum of the two original conductances.



$$i = \frac{v}{R_1} + \frac{v}{R_2}$$

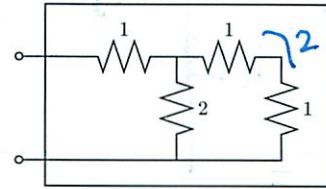
$$i = \frac{v}{R_p}$$

$$\frac{1}{R_p} = \frac{1}{R_1} + \frac{1}{R_2} = \frac{R_1 + R_2}{R_1 R_2} \rightarrow R_p = \frac{1}{\frac{1}{R_1} + \frac{1}{R_2}} = \frac{R_1 R_2}{R_1 + R_2} \equiv R_1 || R_2$$

The resistance of a parallel combination is always smaller than either of the original resistances.

Check Yourself

What is the equivalent resistance of the following circuit.

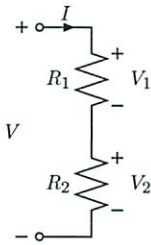


1. 1 2. 2 3. 0.5 4. 3 5. 5

$$\frac{2 \times 2}{2+2} = 1 + 1 = 2$$

*formal way***Voltage Divider**

Resistors in series act as voltage dividers.



$$R = R_1 + R_2$$

$$V = IR$$

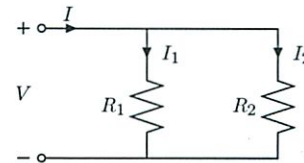
$$I = \frac{V}{R_1 + R_2}$$

$$V_1 = R_1 I = \frac{R_1}{R_1 + R_2} V$$

$$V_2 = R_2 I = \frac{R_2}{R_1 + R_2} V$$

*Want to control what system does**Current same***Current Divider**

Resistors in parallel act as current dividers.

*Voltage same*

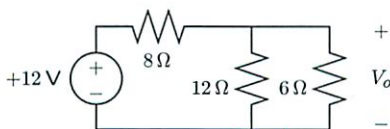
$$V = (R_1 || R_2) I$$

$$I_1 = \frac{V}{R_1} = \frac{R_1 || R_2}{R_1} I = \frac{1}{R_1} \frac{R_1 R_2}{R_1 + R_2} I = \frac{R_2}{R_1 + R_2} I$$

$$I_2 = \frac{V}{R_2} = \frac{R_1 || R_2}{R_2} I = \frac{1}{R_2} \frac{R_1 R_2}{R_1 + R_2} I = \frac{R_1}{R_1 + R_2} I$$

*ratio of that resistance over sum***Check Yourself**

Find V_o .



$$\frac{12 \times 6}{12+6} = 4$$

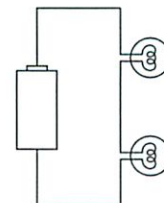
$$V_o = \frac{4}{4+8} \cdot 12V$$

$$\frac{1}{3} \cdot 12V = 4V$$

Loading

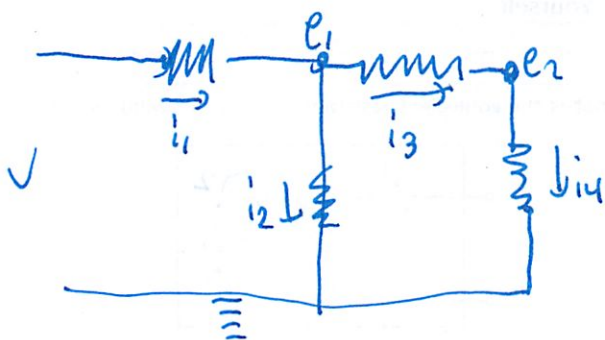
Adding (or changing the value of) a component alters all of the voltages and currents in a circuit (except in degenerate cases).

Consider identical light bulbs connected in series across a battery.



Because the same current passes through both light bulbs, they are equally bright.

formal way



$$i_1 = i_2 + i_3$$

$$i_3 = i_4$$

$$V - e_1 = 1 \cdot i_1$$

$$e_1 - e_2 = 1 \cdot i_3$$

$$e_2 - 0 = 1 \cdot i_4$$

$$e_1 - 0 = 2 \cdot i_2$$

$$V = iR$$

$$V = i_1 + e_1 \\ = i_1 + 2i_2$$

$$e = e_2 + i_3 \\ = i_4 + i_3$$

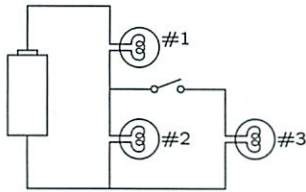
$$2i_2 = 2i_3$$

$$i_2 = i_3$$

$$V = i_1 + 2i_2 \\ = i_1 + i_2 + i_3 \\ = 2i_1$$

Check Yourself

What happens if we add third light bulb?



Closing the switch will make

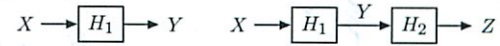
1. bulb 1 brighter
2. bulb 2 dimmer
3. 1. and 2.
4. bulbs 1, 2, & 3 equally bright
5. none of the above

See back

Loading

Loading did not occur in LTI systems.

Example: adding H_2 has no effect on Y



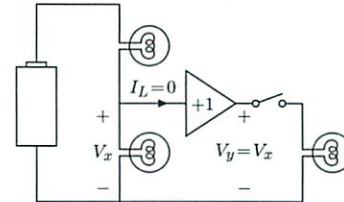
$Y = H_1 X$ regardless of H_2 .

Buffering

Effects of loading can be diminished or eliminated with a buffer.

An "ideal" buffer is an amplifier that

- senses the voltage at its input **without** drawing any current, and
- sets its output voltage equal to the measured input voltage.

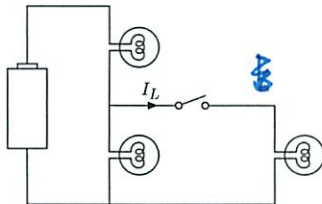


We will discuss how to use op-amps to make buffers in next lecture.

Much more abstract system

Loading

Q: What's different about a circuit?



A: A new component generally alters the currents at the nodes to which it connects.

Changes components other

Summary**Primitives:**

- resistor
- voltage source
- current source

Means of Combination:

- wiring

Means of Abstraction:

- equivalent resistances

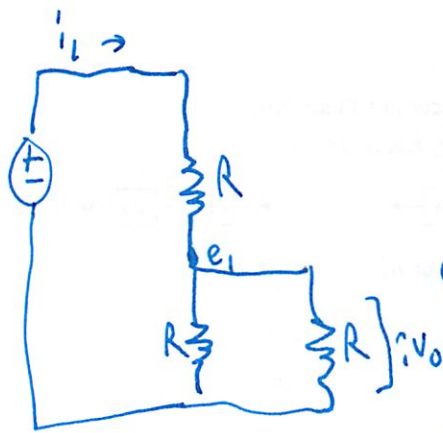
Common Patterns:

- series resistances
- parallel resistances
- voltage dividers
- current dividers

This Week

Software lab: Circuits in simulation and real life

Design lab: Designing photo-detector circuit; controlling robot based on light values



$$i_1 = i_2 + i_3$$

$$V_1 - e_1 = i_1 R$$

$$e_1 - 0 = i_3 R$$

$$e_1 - 0 = i_2 R$$

$$\downarrow$$

$$V_1 = e_1 + i_1 R$$

$$= (i_1 + i_2) R$$

$$i_2 = i_3$$

$$i_1 = i_2 + i_3$$

$$i_1 = 2i_2$$

$$\downarrow$$

$$V_1 = 3 i_2 R$$

$$V_0 = i_2 R = \frac{1}{3} V_1$$

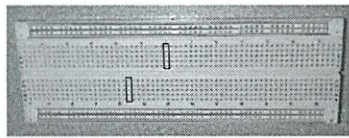
Software (Actually, Hardware)

Lab 7: Divide et impera

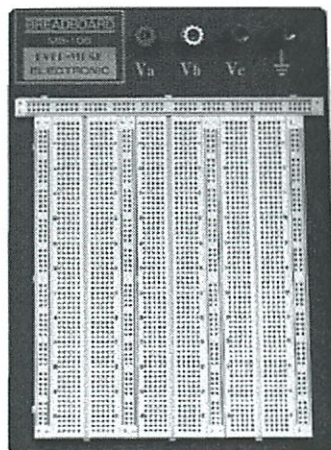
Work with a partner of your own choosing.

- **Using a lab laptop or desktop machine:** Log in using your Athena user name and password; in the terminal window, type `athrun 6.01 update`. Work in the directory `Desktop/6.01/lab7/swLab`.
- **Using your own laptop:** Download the zip file for software lab 7 from the *Calendar* tab of the course web page, unzip it, and work in the resulting directory.

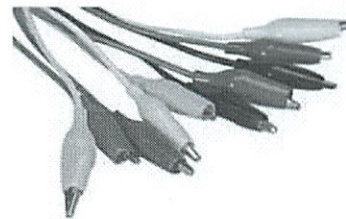
You will also need:



Proto board



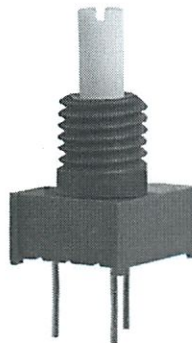
Power supply



Four clip leads



Multimeter



Potentiometer



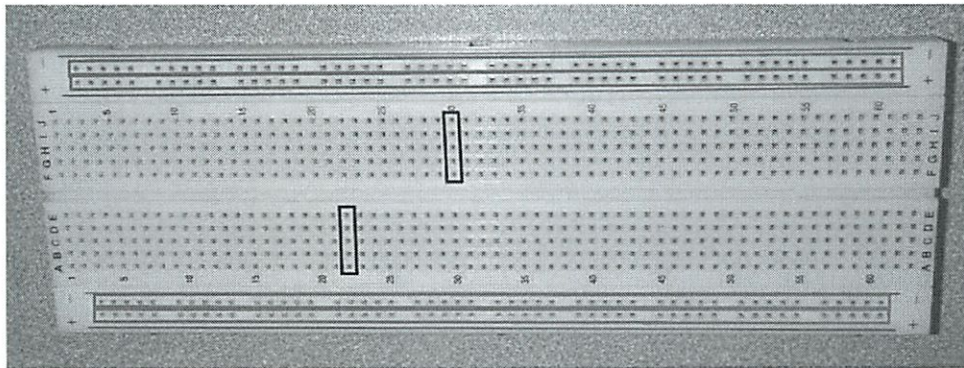
Four 1k Ω , one 100 Ω ,
and one 10K Ω resistor

1k Ω = (brown, black, red); 100 Ω = (brown, black, brown); 10K Ω = (brown, black, orange)

1 Getting started

In today's lab you will build some simple circuits and start learning how to use the 6.01 circuit simulator CMax.

We will build our circuits on a "proto board" like the one shown below.



The proto boards have holes into which wires and components can be inserted. Each column of 5 holes in the center areas is connected internally, as indicated by two representative vertical boxes (above). In other words, if you insert one end of one component into one of the holes in a column of 5, and then insert one end of a separate component into a second hole in the same column of 5, these two components are now connected by an internal wire that connects the 5 holes together. Holes in the top row (marked here with a red line, but sometimes indicated with blue instead) are connected internally (as are those in the second row, bottom row and next-to-bottom row), as indicated by the long horizontal boxes (above). These rows are convenient for distributing power (we will typically use +10 V) and ground.

*Use a **single, separate, proto board** as shown above for this lab. Do not use the proto boards that are attached to the power supply!!!*

It is conventional to use the top rail (which can be either red or blue) for positive supply voltages and the next rail (which can be either red or blue) for ground. Notice that the highest numbered column is on the right and the lowest is on the left.

Step 1.

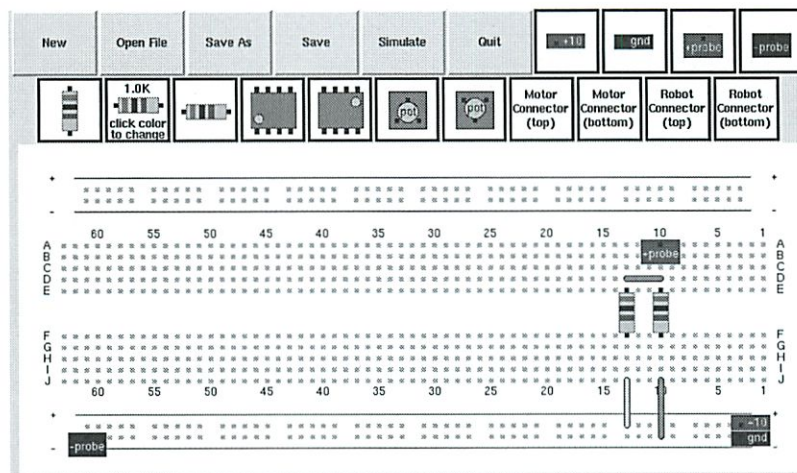
- Connect the power supply terminals labeled +15V and GND to the power rails of your separate proto board using alligator clip leads. Connect the alligator clip to the proto board through short (less than 1") wires (from a wire kit); connect it to the terminal on the power supply by just sticking one 'jaw' of the alligator clip into the center of the terminal. **Don't unscrew the power-supply terminals.**
- Set the multimeter to measure voltage, and connect its probes to the power rails of the proto board using alligator clip leads, through short wires from a wire kit.
- Now, turn on the power supply and measure the power supply voltage with your multimeter. Adjust the positive supply to +10V. This step ensures that your setup will be delivering an appropriate amount of voltage to your protoboard.

2 Layout and Simulation

We will be using a simple layout and simulation tool, called *Circuits Maximus*, or “CMax” to its friends, to design and test circuits before constructing them. The following figure shows a screenshot of CMax. You can run CMax by going to a Terminal window, navigating to lab7/swLab and typing

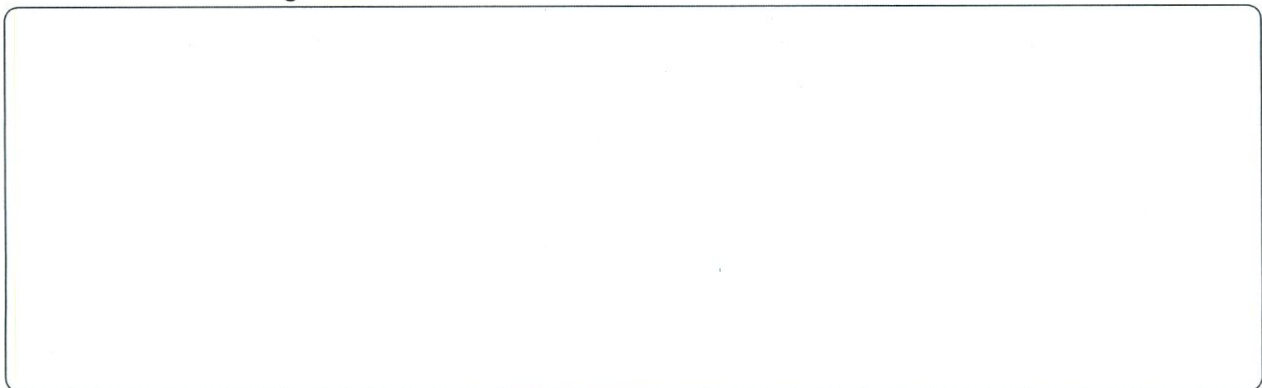
```
> python CMax.py
```

Alternatively, you can start Idle, open the file CMax.py and do Run Module. If you open CMax.py through Idle, close it down by killing Idle.



Step 2. Once CMax is running, press the **Open File** button to open the file `mystery.txt`.

Step 3. Draw a schematic diagram for the circuit shown in the CMax window.

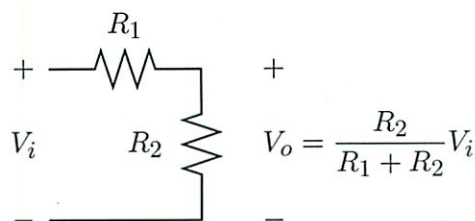


Step 4. Note the meter ‘probes’, attached to the ground rail and to location A10. Predict the voltage that will be measured across those two nodes in the circuit.

Step 5. Press the **Simulate** button (and choose the file `lib601/noInput.py`) to make the simulator calculate V_0 . The result will be printed in the window from which you started Python. Do your calculations match the simulation?

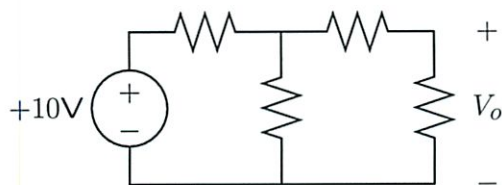
3 Resistor Dividers

A voltage divider is a circuit that uses resistors to generate an output voltage that is a fixed fraction of the input voltage. The following figure illustrates a voltage divider as well as the resulting relation between its input voltage V_i and output voltage V_o :



If $R_1 = R_2$, then $V_o = \frac{V_i}{2}$.

Can we cascade two divide-by-two circuits to produce a divide-by-four circuit? Consider the following design, where all of the resistors have $1\text{ k}\Omega$ resistance.



Step 6. Lay out this circuit using CMax. There is documentation at the end of this handout about using CMax, which you should consult in order to understand how to effectively place components in the simulator.

Try to make your layout simple and clear. Use short wires oriented horizontally or vertically where possible. Try to avoid crossing wires, and do not run wires across other components! You will be using your layout as a guide to constructing a physical circuit. Jumbled wires are more difficult to construct correctly, and they are extremely difficult to debug!

Connect the “probes” so that they measures V_o . Press the **Simulate** button to measure V_o . The value of V_o will be typed in the window from which you started the program.

Check Yourself 1. What is the simulated value of V_o ?

$V_o =$

2.0

Check Yourself 2. Calculate V_o using circuit theory.

$V_o =$

Compare your result to that of the simulation.

Step 7. Lay out your circuit with physical parts. Make your physical layout look **exactly** like the CMax version. Trim the resistor leads so that the resistors lay flat against the proto board. (This step makes it easier to debug your circuit, by presenting a clean view of your components and their layout.)

Check Yourself 3. Measure V_o with your multimeter. Do you get the same voltage V_o ? Exactly?

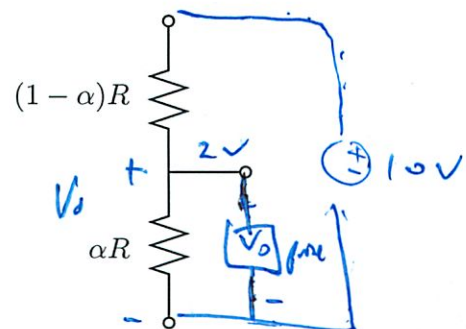
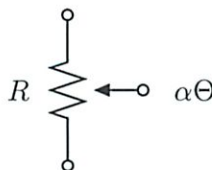
Checkoff 1.

A voltage divider with equal resistors produces an output voltage that is half the value of the input voltage. However, two voltage dividers connected in cascade do not produce an output that is one quarter of the input voltage. Explain why.

Show that your circuit looks **exactly** like the layout in CMax. Show the results from your circuit and simulation.

4 Potentiometer

A potentiometer (or pot) is a three-terminal device whose electrical properties depend on the angle of its mechanical shaft. The following figure shows a picture of the pot that we will use in lab (left), the electrical symbol used for a pot (center), and an equivalent circuit (right). The quantity α is in the range $[0, 1]$; Θ is the maximum turn angle of the pot, for example, 270° , and $\theta = \alpha\Theta$ is the actual turn angle of the pot's input shaft.



$$V_o = \left(\frac{\alpha R}{\alpha R + (1-\alpha)R} \right) V_{in} = \alpha V_{in}$$

As the angle θ of the input shaft increases, the resistance between the bottom and middle terminals increases and the resistance between the middle and top terminal decreases. These changes in resistance occur such that the sum of the top and bottom resistors is constant. If you are interested in the internal construction of a potentiometer, look it up on a source such as Wikipedia. For our purposes, we will treat a potentiometer as a primitive element, which behaves as we just described.

By connecting a pot as a voltage divider (top terminal to a voltage source and bottom terminal to ground), the voltage at the middle terminal is made proportional to the angle of the shaft.

The pots we handed out in lab today have a total resistance of 5K Ω .

- Step 8.** On your protoboard, wire a potentiometer to a 10 V supply and ground. Notice that the leads are arranged in a triangle, with the base of the triangle parallel to one of the straight sides of the pot and the apex of the triangle near the middle of the opposite side. Connect power and ground to the two leads on the base of the triangle.

What are the min and max voltages at the middle terminal of the potentiometer?

min 0V max 10V

- Step 9.** Adjust the potentiometer (the one you just put on your protoboard, **not** the knob on the power supply) so that the voltage on the middle terminal is 2.0V.

To what value of α does this correspond?

$\frac{1}{5}$

- Step 10.** Leaving the pot adjusted as it was in step 9, attach a 100 Ω resistor between the middle terminal of the potentiometer and ground. Measure the voltage V_o at the middle terminal.

1.22 V

- Step 11.** Use circuit theory to compute the ideal value of V_o in this circuit.

1.22 V

- Step 12.** Leaving the pot adjusted as it was in step 9, remove the 100 Ω resistor and attach a 10K Ω resistor between the middle terminal of the potentiometer and ground. Now what is the voltage V_o at the middle terminal?

1.85V

Step 13. Use circuit theory to compute the ideal value of V_o in this circuit.

1.85 V



Wk.7.1.2

Complete this tutor problem based on your computations above.

CMax Documentation

Adding components and wires

- You can place components on your board (resistors, op-amps, etc.) by clicking the associated button. They will appear in the lower left corner of the board, and then you can drag them to where you want them to appear on the circuit. Note that CMax allows you to place components in locations on the board that don't have any holes; this is to give you room to maneuver, but be careful not to leave any components disconnected.
- To obtain a component in a different orientation, hold down the **Shift** key when you select it from the menu.
- Resistors are rectangles with three color bands. You can change the value of the next resistor you place by clicking on the color stripes of the *prototype* resistor icon (the one with the text in it); it will cycle through the colors. **Shift**-clicking a resistor band will cycle through the values in the other direction.
- You can connect any two locations on the board with a wire by clicking on the first spot and dragging to the second.
- Wire ends and component leads must be at one of the gray dots that represent a hole. Only one wire or component lead can occupy a hole.
- You must connect your board to power and ground by adding the +10 and gnd components to it. You must have exactly one of each.

Modifying your circuit

- You can delete a component or wire by holding down the control key (you see a skull/crossbones cursor) and then clicking on the body of the component or on a wire.
- You can move the endpoint of a wire by clicking and dragging it; you can move a whole wire by dragging the middle.
- Moves of components and component creation can be undone using **Undo**. The undo operation is only one level deep, so hitting **Undo** again will re-do the operation.
- To read the value of a resistor in your layout (in case you forget what the color bands mean), **shift**-click the resistor. The value of that resistor will be shown in the prototype resistor button.

File management

- The **Quit**, **Save**, **Save As**, **New** and **Open File** commands should do what you expect. Make sure that the files you create to save your circuits have a `.txt` extension. The **Revert** button will erase the changes you have made since the last time you saved the file.

Running tests

There are several ways to see what happens when you run your circuit. The **Simulate** button will run your circuit; it needs to use an input file that specifies time sequences of inputs to the potentiometers in your circuit. You won't ever need to write an input file; we will specify them for you, to run particular tests.

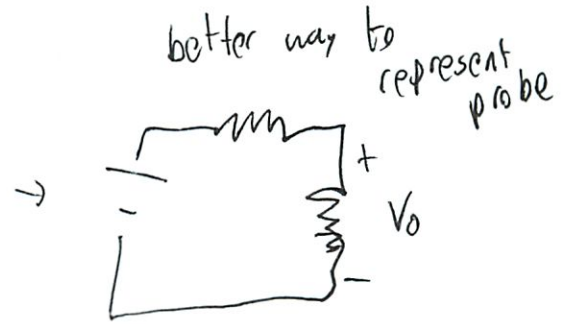
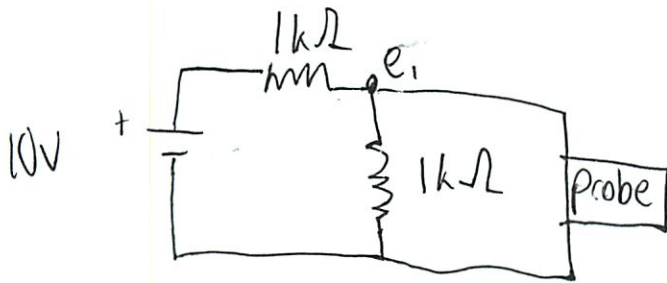
When you click **Simulate** the first time, you pick a test file. It will use the same test file thereafter. If you **Shift-click Simulate**, it will re-prompt you for a test file, so you can select a different one.

- You can measure the voltage between two points in your circuit by placing a +probe and a -probe (exactly one of each) in your circuit, hitting the **Simulate** button, and selecting the file `noInput.py`; it will print the voltage across the probed locations in the window from which you started Python. If there is a component with temporal dynamics (a potentiometer or a motor) in your circuit, then when you simulate, it will also pop up a window showing the signal generated at the probe.
- If there is a *motor* in your circuit, when you hit **Simulate**, a window will pop up that shows a time sequence of the motor's speed, in radians per second.
- If you want to see how your circuit behaves as a function of an input signal, you can add a *potentiometer*. If there is a potentiometer in your circuit, when you hit **Simulate**, a window will pop up that shows a time sequence of the potentiometer alpha values, so you can see what the input is that your circuit is reacting to.

Debugging

Here are some common problems:

- **Failed to solve equations!** Check for short circuit or redundant wire This can be caused by connecting power to ground, for example. Examine your wiring. Maybe you inadvertently used the same column of holes for two purposes. At worst, you can systematically remove wires until the problem goes away, and that will tell you what the problem was.
- `Element ['Wire', 'b47', 'b41'] not connected to anything at node b41` The name 'b41' stands for the bottom group of five holes, in column 41. If you get a message like this, check to see what that element should have been connected to. You know that there should be something else plugged into the bottom section of column 41, in this case.
- **Illegal pin** means that you have a wire or component that has an end or a pin in position on the board that does not have a hole.



$$V = R_1 i + R_2 i \quad i = i_1 = i_2$$

$$10 = 1k\Omega i + 1k\Omega i$$

$$i = 5A$$

$$V - e_1 = i_1 R_1$$

$$e_1 - 0 = i_2 R$$

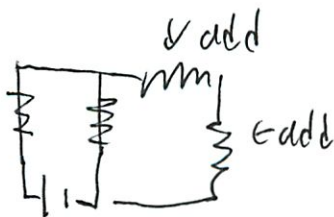
$$e_1 = 5A \cdot 1k\Omega$$

$$e = 5V \quad \textcircled{1} \text{ results match}$$

in window From which started py

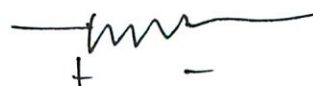
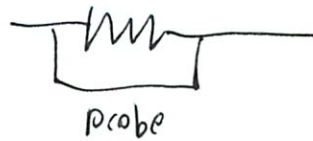
err

3. Resistor Derides

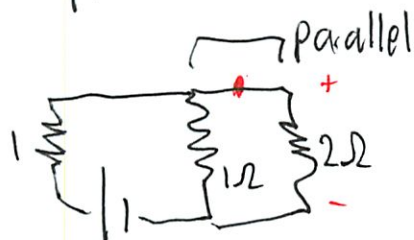
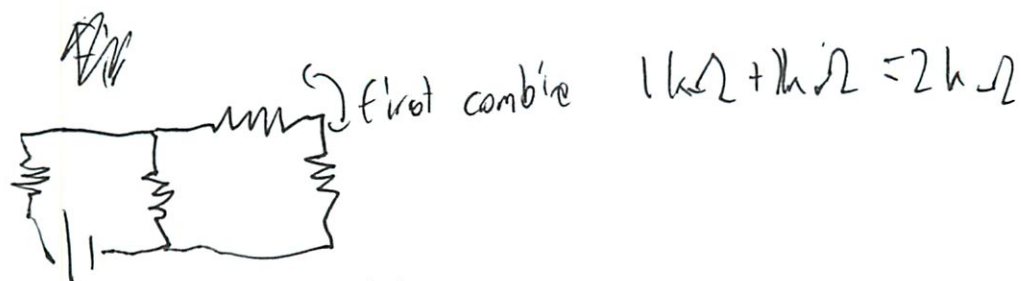


2.0

* make sure probe positioned properly



(2)



$$V = (R_1 \parallel R_2) I$$

$$I_1 = \frac{V}{R_1} = \frac{R_2}{R_1 + R_2} I$$

$$I_2 = \frac{V}{R_2} = \frac{R_1}{R_1 + R_2} I$$

actually more importantly now

parallel $\frac{201}{2+1} = \frac{2}{3}$ equiv resistance

w/ series $\frac{2}{3} + 1$

$$R_{eq} = \frac{5}{3} k\Omega \checkmark$$

$$V = IR$$

$$10 = I \cdot \frac{5}{3}$$

$$I = \frac{10}{\frac{5}{3}} = 10 \cdot \frac{3}{5}$$

$$I = 6$$

not really all that valuable

Now go back

$$I_2 = \frac{V}{R_2} = \frac{1}{1+2} \cdot 6 = 2$$

③

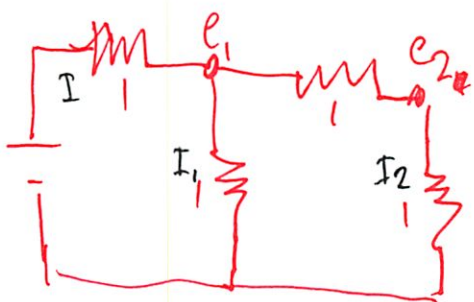
~~$$I_2 = \frac{V}{R_2}$$~~

~~$$2 = \frac{V}{2}$$~~

~~$$V = 4$$~~

try again have to do that first resistor

$$I = I_1 + I_2$$



$$V - e_1 = IR$$

$$e_1 - 0 = I_1 R$$

$$e_2 - 0 = I_2 R$$

$$e_2 = e_1 - I_2 R$$

now start subbing in

~~$$e_2 = e_1 - I_2 R$$~~

$$I = 3I_2$$

$$e_1 - I_2 R = I_2 R$$

$$e_1 = 2I_2 R$$

$$2I_2 = I_1$$

$$10 - 2I_2 R = 3I_2 R$$

$$10 = 5I_2 R$$

$$10 = 5V$$

$$V = 2$$

13(4)

- in parallel \rightarrow can't just add stuff

4. α = degree of turn

θ = max turn

$\theta = \alpha \theta$ actual angle

step 8

min = 0 V) assuming turns through whole range
max = 10 V

but are $5k\Omega$

$$V_i = IR$$

$$10 = I \cdot 5k\Omega \quad \text{max}$$

$$I = 2A$$

so what does that mean?

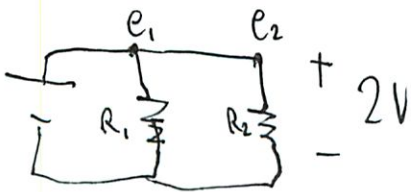
~~10V~~

$$V_0 = 2 \cdot 5k\Omega$$

$$V_0 = 10V$$



step 9



~~10V~~

$$V - e_1 = 0$$

$$V e_1 - 0 = I_1 R_1$$

$$e_2 - 0 = I_2 R_2 = 2$$

* One is \propto , one is $\propto 1 - \alpha$

Calculate R_{eq} ?

But don't know resistance

5

are told $R_1 = 5k\Omega$

R_2

} sum together get 5Ω

$$R_1 = (1-\alpha) \cdot 5k\Omega$$

$$R_2 = 2.5k\Omega$$

$$e_1 - 0 = I_1 (1-\alpha) \cdot 5$$

$$e_2 - 0 = I_2 \cdot \alpha \cdot 5 = 2$$

$$R_{eq} = \frac{\cancel{I_1} (1-\alpha) \cdot 5 \cdot \cancel{I_2} \alpha \cdot 5}{\cancel{I_1} (1-\alpha) \cdot 5 + \alpha \cdot 5}$$

$$\frac{25(1-\alpha)\alpha}{5(\alpha + 1-\alpha)}$$

$$5(\alpha + 1-\alpha)$$

$$\frac{25\alpha - 25\alpha^2}{5}$$

$$R_{eq} = 5\alpha - 5\alpha^2$$

$$V = IR$$

$$10 = I + (5\alpha - 5\alpha^2)$$

$$I = \frac{10}{5\alpha - 5\alpha^2} = \frac{2}{\alpha - \alpha^2} = \frac{2}{\alpha(1-\alpha)}$$

6

$$I_1 = \frac{V}{R_1} = \frac{R_2}{R_1 + R_2} I$$

$$\frac{5\alpha}{5} \cdot \frac{2}{\alpha(1-\alpha)} = \frac{V}{5(1-\alpha)}$$

we want

$$I_2 = \frac{V}{R_2} = \frac{R_1}{R_1 + R_2} I$$

$$\frac{\cancel{5}(1-\alpha)}{\cancel{5}} \cdot \frac{2}{\cancel{2}(1-\alpha)} = \frac{V}{5\alpha}$$

$$\frac{2}{\alpha} = \frac{\cancel{2}V}{5\alpha}$$

$$10\alpha = V\alpha$$

$$10 = V$$

Confirmed we have 10 V

but told we have 2 V, need α

$$\frac{2}{5\alpha} = \frac{2}{\alpha}$$

$$2\alpha \neq 10\alpha$$

~~Wrong~~ Wrong

⑦

Another try

$$10 = I R$$

$$10 = I 5 \alpha - 5 \alpha^2$$

$$I = \frac{2}{\alpha(1-\alpha)} \quad \text{found this already}$$

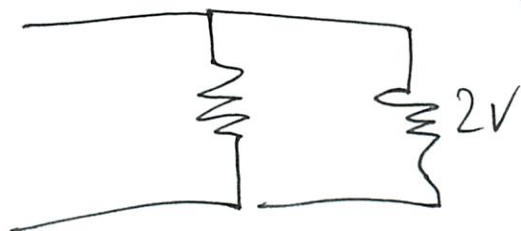
$$I = I_1 + I_2$$

$$\text{or just } \frac{V_1}{V_2} = \frac{2}{8} = \frac{1}{4}$$

$$\alpha = 1/4$$

$$\Theta = 270^\circ$$

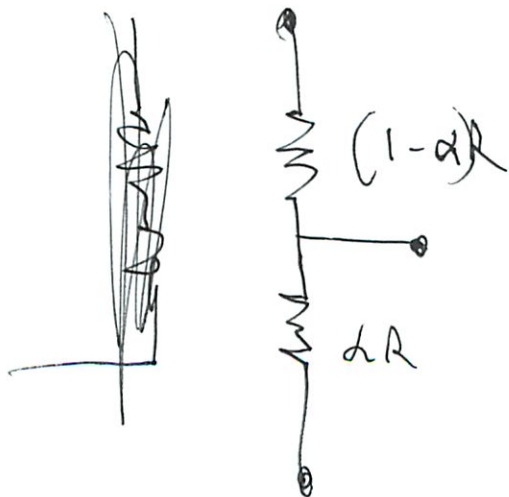
← simple



← current divider

↳ Voltage same
↳ parallel

Want voltage divider ← current same among both



$$2V = I_{\text{system}} \alpha R$$

need R_{eq} , find I_{system}

$$\textcircled{8} \quad V_o = \frac{\alpha R}{\alpha R + (1-\alpha)R} = V_2 = \frac{R_2}{R_1 + R_2} V$$

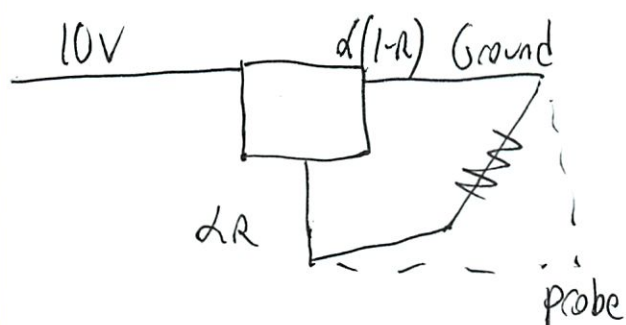
$\frac{2}{8}$ does not matter here

$$V_o = \alpha V_{in}$$

$$2 = \alpha 10$$

$$\alpha = \frac{1}{5}$$

Which is αR ?



Step 10

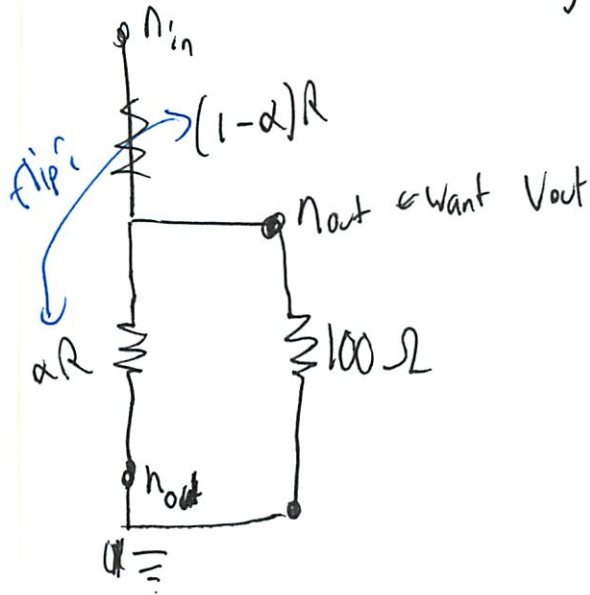
22 V
draw did

9

Step 11

10/20
Home

$$\alpha = \frac{1}{5}$$



$$V_o = I_{system} \alpha R$$

$$V_o = I_{sw} \frac{1}{5} 100 \Omega$$

is this it?

not doing displaced

-copying last av

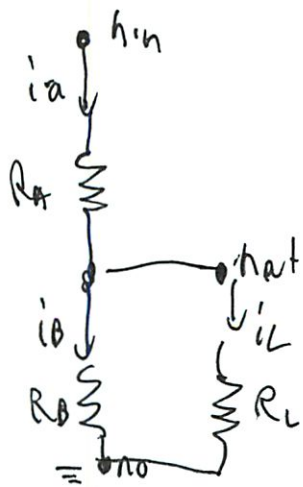
-try it quick

$$V_o = \frac{\frac{1}{5} R}{\frac{1}{5} R + \frac{4}{5} R}$$

$$V_o = \alpha V_{in}$$

$$V_o = \alpha 10$$

but this does not concern me
start over w/ course notes



$$R_a = \alpha = \frac{1}{5}$$

$$R_b = 1 - \alpha = \frac{4}{5}$$

$$R_c = 100 \Omega$$

$$V_{in} = 10 V$$

$$V_{out} = ?$$

(10)

So ~~not~~ NVCC $n_o = \text{ground}$

$$V_o = 0$$

KCL = not much of a loop

$$i_B = i_A - i_{out}$$

$$V_{in} = 10 \quad V_o = 0 \quad V_{out} = i$$

$$V_{in} - V_{out} = i_A R_A$$

$$V_{out} - V_o = i_B R_B$$

$$V_{out} - V_o = i_L R_L \quad \text{J is correct is series}$$

So

$$R_{eq} = \frac{100 \cdot \frac{4}{5}}{100 + \frac{4}{5}} = \frac{80}{100.8}$$

No should be able to go w/o

$$\frac{V_{out} - V_o}{R_B} = \frac{V_{in} - V_{out}}{R_A} - \frac{V_{out} - V_o}{R_L}$$

$$\frac{V_{out}}{R_B} = \frac{10 - V_{out}}{R_A} - \frac{V_{out}}{R_L}$$

$$\frac{(10 - V_{out}) R_L - V_{out} R_A}{R_A - R_L}$$

(11)

$$V_{out} = \cancel{R_B R_2 \cdot 10} - \cancel{R_B R_L} V_{out}$$

Need to collect V_{out}

Or put in resistor $\#$ to simplify

$$\frac{V_{out}}{\frac{4}{5} \cdot 5k\Omega} = \frac{10 - V_{out}}{\frac{1}{5} \cdot 5k\Omega} - \frac{V_{out}}{100\Omega}$$

$$\cancel{11} \frac{V_{out}}{4000\Omega} = \frac{10 - V_{out}}{1000\Omega} - \frac{V_{out}}{\cancel{100}\Omega}$$

$$\frac{V_{out}}{4000} = \frac{10 - V_{out} - 10 V_{out}}{1000\cancel{100}}$$

$$\frac{V_{out}}{4000} = \frac{10 - 11 V_{out}}{1000 \cdot 1000}$$

$$\frac{1000 V_{out}}{4000} = 10 - 11 V_{out}$$

$$\frac{1}{4} V_{out} = 10 - 11 V_{out}$$

$$11 \frac{1}{4} V_{out} = 10$$

$$V_{out} = \frac{10}{11.25}$$

$$= 0.88 V$$

(12) Hmm .88 and .22 related?

test in tutor (x)

Nope

Well course notes says

$$V_{out} = V_{in} \frac{R_B}{R_A + R_B + \frac{R_A R_B}{R_L}}$$

$$= 10 \cdot \frac{4000}{4000 + 1000 + \frac{4000 \cdot 1000}{100}}$$

$$= 10 \cdot \frac{4000}{5000 + 40000}$$

$$= 10 \cdot .088$$

$$= .88$$

same answer wtf - at least answer was right w/ steps

am I flipping R_A and R_B

-try

$$= 10 \frac{1000}{1000 + 4000 + \frac{4000 \cdot 1000}{100}}$$

$$= .22 \text{ so will try that}$$

~~but I tried~~ so I reversed
how know which is which?

21

(13)

Now swap 100 α for 10,000

$$= 10 \frac{1000}{1000 + 4000 + \frac{4000 \cdot 1000}{10000}}$$

$$= 10 \cdot \frac{1000}{5400}$$

$$= 1.85 \text{ (1)}$$

crew was right on w/ his real world testing

I guess which is which depends on what you connect to what

so if $\alpha = \frac{4}{5}$ my $A+B$ would be right

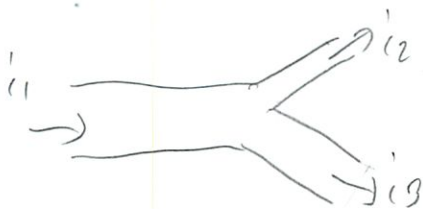
We did find α algebraically - but w/ ^{1st} α !

so that is why we got $\frac{1}{5}$

Circuits Reading

10/20

- incompressible



$$i_1 = i_2 + i_3$$

+ - +
- of +

Kirchoff's
Current Law

Net Flow at every node must be 0

Current that flows into an element must flow out

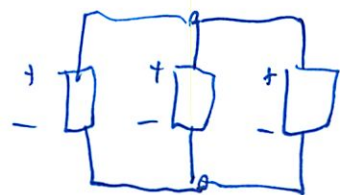
Equations must not be linearly dependent



Kirchoff's Voltage rule

pressure wants to = lize

Sum of voltage differences along closed path is 0



$$\begin{aligned} -V_1 + V_2 &= 0 \\ V_1 &= V_2 \end{aligned}$$

$$\begin{aligned} -V_2 + V_3 &= 0 \\ V_2 &= V_3 \end{aligned}$$

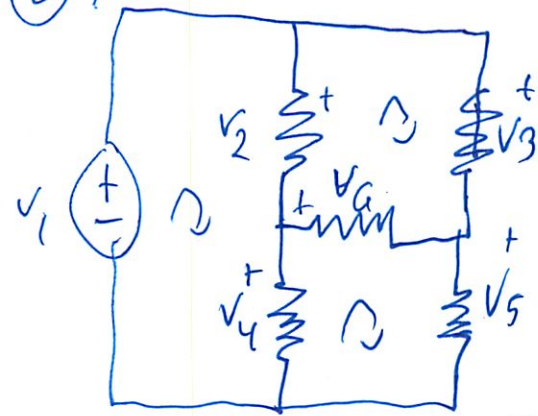
$$\begin{aligned} -V_1 + V_3 &= 0 \\ V_1 &= V_3 \end{aligned}$$

2 eq are linearly independent

once can be found from other 2

$$V_1 = V_2 = V_3$$

② ↻
 what about kirchoff's loop rule?
 - I think that is voltage rule



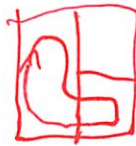
$$\begin{aligned}
 V_1 - V_2 - V_4 &= 0 \\
 -V_3 + V_6 + V_2 &= 0 \\
 V_5 + V_4 - V_6 &= 0 \\
 V_1 - V_3 - V_5 &= 0
 \end{aligned}$$

↗ pay attention!

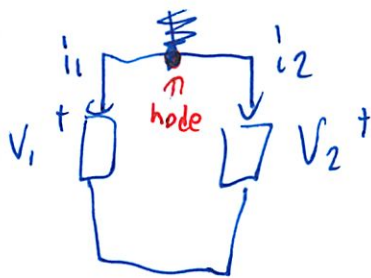
they flipped the signs

they do the sign you come in on - may be better idea

* can also do weirder shapes



7.4 Constitutive Equations



First current laws

$$\begin{aligned}
 i_1 + i_2 &= 0 \text{ at node} \\
 \text{then } V_1 - V_2 &= 0
 \end{aligned}$$

↻

each element has some effect on current + voltage through across

voltage source



(3)

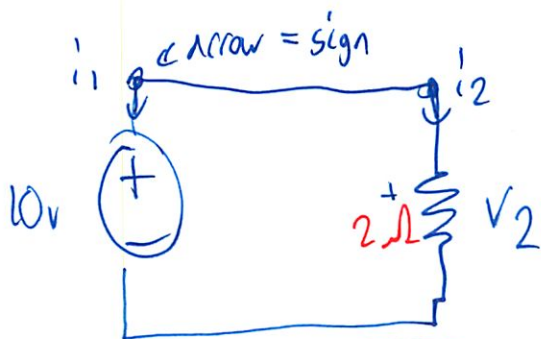
Current Source

$$i = I_0 \quad \text{independent}$$

what are these again
 batt = voltage source + current source
 wall = AC source

Resistor

$$V = iR$$



~~10V~~ \sim

$$-10V + V_2 = 0$$

$$V_2 = 10V$$

$$i_1 = -i_2 \quad \checkmark \quad \neq 0$$

$\begin{matrix} i_1 & i_2 \\ \leftarrow & \rightarrow \end{matrix}$

$$i_1 + i_2 = 0$$

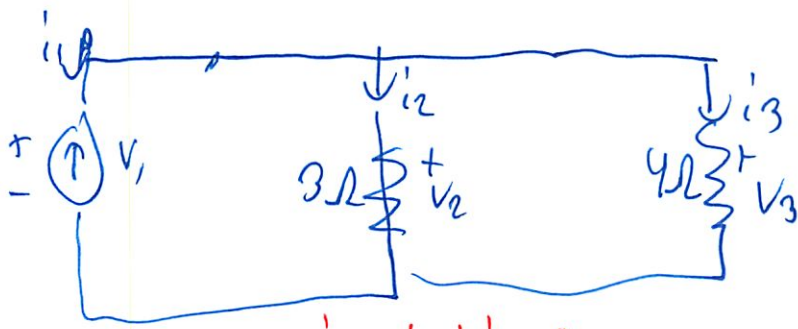
$$i_2 = \frac{V}{2\Omega} = 5A$$

can calc current!

$$i_2 = -5A$$

4

Example 8



$i_1 + i_2 + i_3 = 0$

~~$i_1 = -i_2$ $i_2 = -i_3$~~
 ~~$i_1 = i_3 = -i_2$~~

but this is not true

$-V_1 + V_2 = 0$

$V_1 = +V_2$

$V_3 - V_2 = 0$

$V_3 = +V_2$

$-V_1 + V_3 = 0$

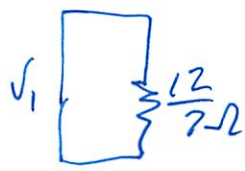
$V_1 = V_3$

$V_1 = V_2 = V_3$

$V_2 = i_2 3\Omega$

$V_3 = i_3 \overset{\text{duh}}{4\Omega}$

$R_{eq} = \frac{3 \cdot 4}{3+4} = \frac{12}{7} \Omega$



missed

$V_1 = i_2 \frac{12}{7} R$

now how to solve
- convert to V_1 i_2

~~missed~~

$i_2 \frac{12}{7} R = i_2 \frac{3R}{3}$

$\frac{12}{7 \cdot 3} = \frac{12}{21} \Omega i_2 = i_2$

does not balance

⑤ Wait yes it does

$$i_2 \left(\frac{12}{21} - 1 \right) = 1$$

$$i_2 = \frac{\cancel{21} 12 - 21}{21}$$

$$= \frac{-9}{21}$$

$$V_2 = \frac{9}{21} \cdot 3$$

$$\frac{21}{21}$$

$$V_2 = V_1 = 1V$$

No - Remember parallel is current divider

And i_1 was given $-14A$

$$V_1 = V_2 = 3\Omega \cdot i_2 = V_3 = 4\Omega \cdot i_3$$

$$3\Omega \cdot i_2 = 4\Omega \cdot i_3 \quad \text{replace for}$$

$$3\Omega \cdot i_2 = 4\Omega (14 - i_2)$$

$$i_2 = 8A$$

$$i_3 / 3 \cdot 8 = 4 \cdot i_3$$

$$i_3 = 6$$

$$V_1 = V_2 = V_3 = 3 \cdot 8 = 24V$$

⑥ So then what was i_2

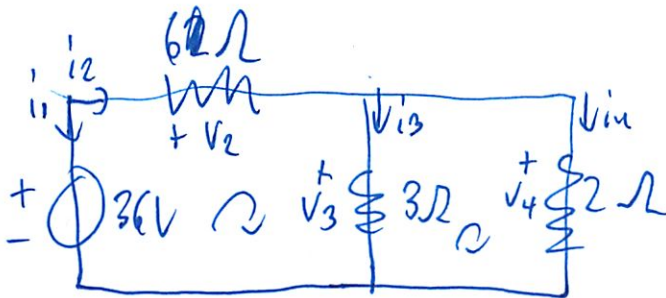
$$i_2 = -i_1 - i_3$$

$$i_3 = -i_2 - i_1$$

$$i_1 = -i_2 - i_3$$

Example 9

← what we did in class



$$i_1 = -i_2$$

$$i_3 = -i_4 + \frac{V_2}{6\Omega}$$

$$i_4 = -i_3 - \frac{V_2}{6\Omega}$$

$$V = iR$$
$$i = \frac{V}{R}$$

$$-36 + V_2 + V_3 = 0$$

$$V_4 - V_3 = 0$$

$$\Rightarrow V_4 = -V_3$$

$$-36 + V_2 + V_4 = 0$$

So where to start?

- no steps in solution

- do similar to last one

$$V_4 = -V_3$$

$$3\Omega i_3 = -2\Omega i_4$$

$$3\Omega i_3 = -2\Omega \left(-i_3 - \frac{V_2}{6\Omega} \right)$$

$$\begin{array}{ccc} 3\Omega i_3 & = & +2\Omega i_3 + \frac{V_2}{6\Omega} \\ -2\Omega i_3 & & -2\Omega i_3 \end{array}$$

$$\cancel{4\Omega} i_3 = \frac{V_2}{6\Omega}$$

$$6\Omega i_3 = V_2 \quad \text{e does not really help us here}$$

$$-36 + 6\Omega i_3 + 2\Omega i_4 = 0$$

$$6\Omega i_3 + 2\Omega i_4 = 36$$

$$i_3 \cancel{6\Omega} = \frac{36 - 2\Omega i_4}{6}$$

$$i_3 = 6 - \frac{1}{3}\Omega i_4$$

$$\frac{V_2}{2\Omega} = 6 - \frac{1}{3}\Omega i_4$$

$$V_2 = 12 - \frac{2}{3}\Omega i_4$$

what is this getting me?

8

Could do more basic

$$i_1 = \frac{V_2}{6\Omega} = -i_2$$

$$i_2 = -\frac{V_2}{6\Omega}$$

mm

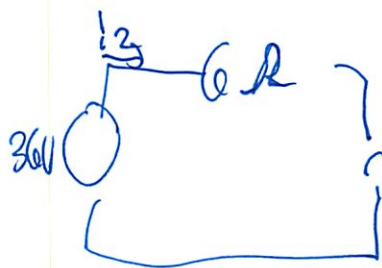
Oh $V_1 = 36V$ -

- why do I always forget this

- no I included it

Look at class notes

perhaps Rev



$$\leftarrow \frac{2+3}{2+3} \frac{6}{5} \Omega$$

$$V = IR$$

$$36 = i_2 \left(6 + \frac{6}{5} \right)$$

$$36 = 7.2 i_2$$

$$i_2 = 5$$

$$i_1 = -5A \quad (\checkmark)$$

Does current change in resistor - no!

$$i_3 = -i_4 + i_2$$

$$i_3 = -i_4 - 5$$

so can I say this

⑨

parallel = voltage divider

cascade = \int current divider \leftarrow so it does?

but can go back

parallel = current dividers
series = Voltage divider

$$i_2 = \frac{-V_2}{2\Omega}$$

$$5 = \frac{-V_2}{2\Omega}$$

$$V_2 = 10V \quad (\otimes) 30V$$

$$-36V + 10V + V_4 = 0$$

$$10V + V_4 = 36V$$

$$V_4 = 26V \quad (\otimes) 6V$$

$$i_4 = 2A$$

$$i_3 = \frac{V_2}{6\Omega}$$

$$= \frac{10V}{6\Omega}$$

$$= \frac{5}{3} A \quad (\otimes) 2A$$

$$i_4 = -\frac{5}{3} - \frac{10}{6}$$

$$i_4 = -\frac{10}{3} \quad (\otimes) 3A$$

$$V_3 = 6V \quad (\otimes)$$

10

$$-36 + 10 + V_4 = 0$$

$$V_4 = -26V \quad \leftarrow \text{did already}$$

$$V_3 = 26V \quad \otimes 6V$$

think that's all

So that was a disaster
but how to do better

try again on own

- confused about w/ G/N

$$i_1 = -i_2$$

- check lecture example

$$i_3 =$$

Well in class they use \mathcal{C}

$$\mathcal{C}_1 - 0 = 3 \mathcal{A} \cdot i_2$$

and the obvious

$$i_2 = i_3 + i_4$$

well that was not obvious

along a path current
is constant

but

So yes current does not change through
a resistor!

11

$$i_1 = -i_2$$

$$i_2 = i_3 + i_4$$

$$i_3 = -i_4 + i_2$$

$$i_4 = -i_3 + i_2$$

I think my \checkmark loops were right

$$-36 + V_2 + V_3 = 0$$

~~$V_4 - V_3 = 0$~~ $\int V_4 = V_3$

$$-36 + V_2 + V_4 = 0$$

And finally $36 = V_1$ I'm there already

$$V = IR$$

$$V_3 = i_3 \ 3\Omega$$

$$V_4 = i_4 \ 4\Omega$$

$$i_3 3\Omega = -(-i_3 + i_2) 4\Omega$$

$$3i_3 = -4i_2 + 4i_3$$

$$-1i_3 = -4i_2$$

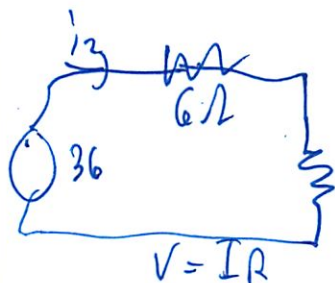
but how move past that

12

Do I need to do beg?

- so many ways to do

- am I taking too many steps?



$$\leftarrow \frac{36}{3+2} = \frac{3 \cdot 2}{3+2} = \frac{6}{5}$$

$$36 = i_2 \left(6 + \frac{6}{5} \right)$$

$$i_2 = 5A \quad \textcircled{1}$$

$$i_1 = -5A \quad \textcircled{1}$$

Now plug back

$$-1 \cdot i_3 = -4(5A)$$

$$i_3 = -20$$

$$i_3 = \frac{20}{1} \quad \textcircled{X} \quad 2A$$

$$\text{But } i_2 = i_3 + i_4$$

Or did I make a sign error before

Wrong as well
or try again

$$i_3 \cdot 3 = - (i_2 - i_3) \cdot 4$$

$$3i_3 = -4i_2 + 4i_3$$

$$3i_3 = -4 \cdot 5 + 4i_3$$

(13)

$$-i_3 = -20$$

$$i_3 = 20 \text{ A } (\times)$$

Did I make mistake w/
base eq

↓ so why does this work now

Oh could get V_2

$$V_2 = 5 \cdot 6$$

$$V_2 = 30$$

$$-36 + 30 + V_3 = 0$$

$$V_3 = 6 \quad (\checkmark)$$

$$V_3 = i_3 \cdot 3$$

$$6 = i_3 \cdot 3$$

$$i_3 = 2 \quad (\checkmark)$$

$$V_4 = +V_3$$

$$V_4 = V_3 = 6 \quad (\checkmark)$$

$$V_4 = i_4 \cdot 2$$

$$6 = i_4 \cdot 2$$


$$i_4 = 3 \quad (\checkmark)$$

done

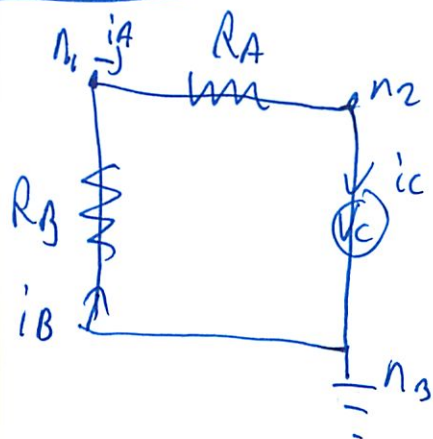
-so what did I do wrong upfront
something w/ i

(1.4)

7.5 Common Patterns

- oh here is where they explain the process
- 1. - associate voltages w/ nodes
 - not components
- 0. - associate a fixed ref voltage
 - pick a node (usually - terminal)
 - call it ground 
 - give $V=0$
- so voltage is voltage difference b/w a point + ground = V

Resistors in Series



$$\begin{aligned}
 i_A - i_C &= 0 && \downarrow \text{current does change over resistor} \\
 i_B - i_A &= 0 \\
 V_3 &= 0 && \leftarrow \text{since we called it ground} \\
 V_1 - V_2 &= i_A R_A && \leftarrow \text{voltage drop over resistor} \\
 V_3 - V_1 &= i_B R_B \\
 V_2 - V_3 &= V_C && \leftarrow \text{the source}
 \end{aligned}$$

15

So how to solve
(finally!)

$$V_3 - V_2 = i_A R_A + i_B R_B = -V_C$$

~~the sum~~ \uparrow the sum of the circuit

$$= i (R_A + R_B) = -V_C$$

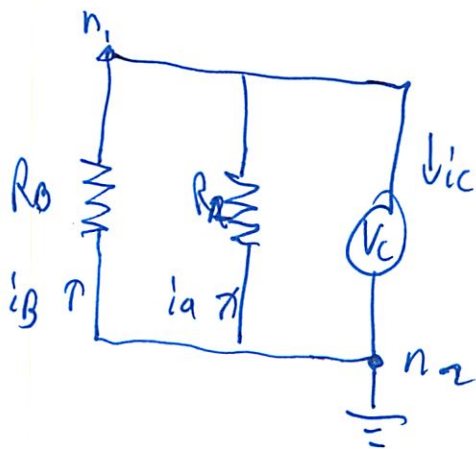
\uparrow $i_A = i_B$

$$-i = \frac{V_C}{R_A + R_B}$$

negative since current flows other direction as arrows

or Req combining resistors

Resistors in Parallel



$$i_A \neq i_B$$

$$n_1 \rightarrow i_A + i_B - i_C = 0$$

$$V_2 = 0 \leftarrow \text{ground}$$

$$V_2 - V_1 = i_A \cdot R_A$$

$$V_2 - V_1 = i_B \cdot R_B$$

~~the~~

$$V_1 - V_2 = V_C$$

(6)

So simplify

$$-V_c = i_A \cdot R_A$$

$$-V_c = i_B \cdot R_B$$

Solve for each:

$$i_A = -\frac{V_c}{R_A}$$

$$i_A \neq i_B$$

$$i_B = -\frac{V_c}{R_B}$$

Now put it all together

$$i_A + i_B - i_c = 0$$

Plug in (like did before)

$$-\frac{V_c}{R_A} - \frac{V_c}{R_B} - i_c = 0$$

$$-i_c = \frac{V_c}{R_A} + \frac{V_c}{R_B}$$

Get common denom

$$i_c = \frac{V_c R_B - V_c R_A}{R_A + R_B}$$

$$= \frac{V_c (R_B + R_A)}{R_A + R_B}$$

$$\text{Solve for } V_c \quad -V_c = i_c \quad \frac{R_A R_B}{R_A + R_B}$$

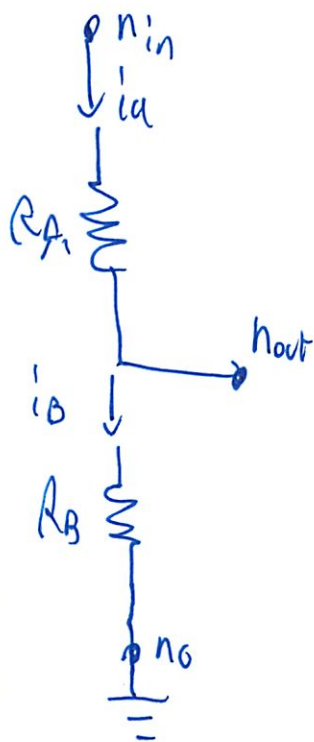
(17)

So notice $\frac{R_A R_B}{R_A + R_B}$

If R_A big, i_A is small

So that same voltage/pressure drop on both

Voltage divider



$$V_{out} = \frac{R_B}{R_A + R_B} V_{in}$$

$$V_0 = 0 \text{ } \leftarrow \text{ground}$$

$$i_A - i_B = 0$$

\therefore does none come out now?

$$V_{in} - V_{out} = i_A R_A$$

$$V_{out} - V_0 = i_B R_B$$

$$V_{in} - 0 = i R_A + i R_B \quad \leftarrow \text{Voltage drop down the line} \quad | -$$

$$V_{in} = i (R_A + R_B)$$

$$i = \frac{V_{in}}{R_A + R_B}$$

(18)

Now try to solve other stuff

~~i_A~~

$$V_{in} - V_{out} = i R_A$$

$\tau_{pkg} i_n$

\downarrow

$$V_{in} - V_{out} = V_{in} \frac{R_A}{R_A + R_B}$$

distribute

$$V_{in}(R_A + R_B) - V_{out}(R_A + R_B) = V_{in} R_A$$

$$V_{in} R_B = V_{out}(R_A + R_B)$$

$$V_{out} = V_{in} \frac{R_B}{R_A + R_B}$$

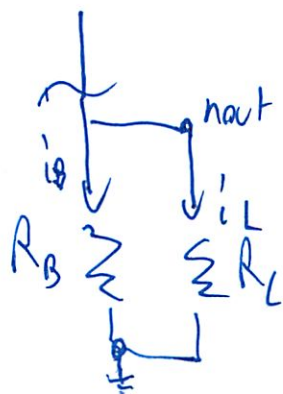
Now if $R_A = R_B$

$$V_{out} = \frac{V_{in}}{2}$$

So if have voltage you need,
can figure out resistance

BUT

Current out V_{out} (what I said before!)



$\hookleftarrow R_L$ is whatever resistance
the additional load must offer

(p)

Makes a big difference

- R_B and R_L in parallel

- like $\frac{R_B R_L}{R_B + R_L}$

- so $V_{out} = V_i \frac{R_B}{R_A + R_B + \frac{R_B R_L}{R_L}}$

Not very modular, but somewhat

7.6 Solving circuits systematically

~~1~~ Solve a set of equations to find voltage across every element and every circuit element

$2n$ # of independent eq needed

n # of elements

n for constitutive relations

n for KVC/KCL laws

- often not the simplest

- also Nodal + node methods

- to ↓ redundant KVL equations

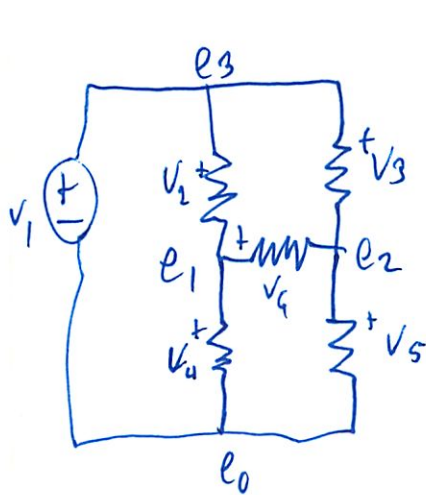
- not do primitive loops

(what 8.02 taught)

(2)

Circuit planer if can draw w/o overlap

↳ not easy to determine
node voltages can make this easier



$$V_1 = e_3 - e_0$$

$$V_2 = e_3 - e_1$$

$$V_3 = e_3 - e_2$$

$$V_4 = e_1 - e_0$$

$$V_5 = e_2 - e_0$$

$$V_6 = e_1 - e_2$$

~~this~~ by substituting node voltages for element voltages
remove redundancy

- but new redundancy of global offset voltage

↳ arbitrarily assign a node to 0 → ground
(how is this different?)

NVCC

Node voltage and component current

Simpler but long winded

good for computers

(21)

1. Label all node n_1, n_2, \dots
Give a voltage at each V_1, V_2, \dots
2. Declare 1 to be a ground node
 - can be anything
 - set $V_g = 0$
3. Make current variables for each component
 - pick a direction + use it consistently
4. Write down $n-1$ KCL equations
 - for each except n_g
 - sum of currents entering each node $= 0$
5. Write m constitutive equations, one for each component
 - with current i_k
 - and voltage $V_{k+} - V_{k-}$
 - \uparrow positive terminal
 - \uparrow negative terminal
 - direction of current defines + / -
 - current runs $(+) \rightarrow (-)$
6. Solve equations

(22)

So Resistor

$$V_{k+} - V_{k-} = i_k R$$

Voltage source

$$V_{k+} - V_{k-} = V_s$$

Current source

$$i_k = I_s$$

Strategy for humans

- to ↓ brute force

- try to eliminate unneeded current variables

$$i_A - i_B - i_D - i_C = 0$$

$$i_B - i_A = 0$$

Sub in constitutive eq

$$\frac{V_1 - V_2}{R_A} - \frac{V_2 - V_1}{R_B} - \frac{V_C}{R_D} - i_C = 0$$

$$\frac{V_2 - V_1}{R_B} - \frac{V_1 - V_2}{R_A} = 0$$

Now have 2 eq and 2 unknowns (V_1, i_C)

We know $V_C = V_1$

$$i_C = \frac{V_C}{R_D}$$

Should try some problems - but later

(note not copying
whole example)

(23)

Node analysis

instead 1 eq per node

Since an element constrains that the 2 node voltages can be taken from one another

if circuit has n nodes, need $n-1$ independent KCL eq w/ $n-1$ node voltage variables

if more than 1 voltage source 2 issues
- skipping for now

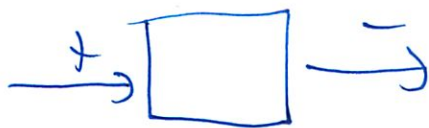
7.7 Circuit Equivalents

can ~~still~~ abstract to constraints

- since can't abstract completely

if only resistors, current, + voltage sources

then we can make a much simpler circuit
(Req right?)



↑ 'whats in the box'

- look at open circuit voltage (V_{oc})
↳ if nothing was connected

(24)

If the wires would be connected directly together

↳ Short circuit current i_{sc}

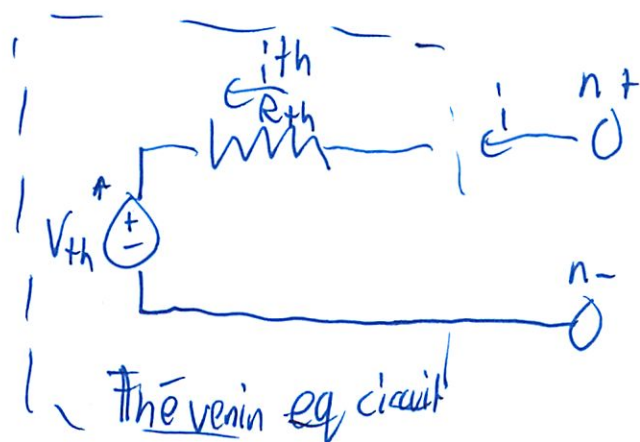
Since relationships are linear, we know what box will do to circuit

* Thevenin's Theorem \Rightarrow any combo of current + voltage sources + resistors can be replaced w/
Single voltage source V_{th} and single resistor R_{th}

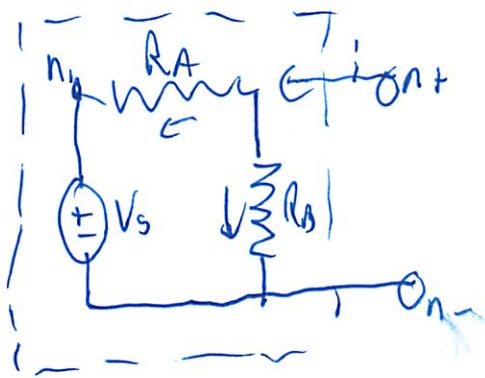
$$V_{th} = V_{oc}$$

$$R_{th} = \frac{V_{th}}{-i_{sc}}$$

~~There~~



(25) So if actual circuit looked like



First compute open circuit voltage

- diff $n_+ \rightarrow n_-$ assuming $i=0$
- set n_- as ground
- find n_+

$$V_+ - V_1 = i_A R_A$$

$$V_1 - V_- = V_s$$

$$V_+ - V_- = i_B R_B$$

$$i_A - i_B = 0$$

$$i_A - i_s = 0$$

$$V_- = 0$$

3 sections

nodes

(26)

Solve for V_t

$$V_t - V_1 = i_A R_A$$

$$V_1 - 0 = V_S$$

$$V_t - V_S = i_A R_A$$

$$i_B R_B - V_S = i_A R_A$$

$$-V_S = i_A R_A - i_B R_B$$

ah solve for V_t

~~$$-V_S = i_A R_A$$~~

$$V_t = i_A R_A + V_S$$

$$V_t = i_B R_B + V_S$$

$$V_t = \frac{V_t R_A}{R_B} + V_S$$

$$V_t - \frac{V_t R_A}{R_B} = V_S$$

$$V_t \left(1 - \frac{R_A}{R_B}\right) = V_S$$

$$V_t = \frac{V_S}{1 - \frac{R_A}{R_B}}$$

why am I
not getting
their "obvious"
answer

$$(20) \quad V_+ = V_S \cdot \left(1 - \frac{R_B}{R_A}\right)$$

$$V_+ = V_S - \frac{V_S R_B}{R_A}$$

$$V_+ = V_S \cdot \frac{R_B}{R_A + R_B}$$

Start over "straight forward"

$$V_+ - V_1 = i_A R_A$$

$$V_1 - 0 = V_S$$

$$V_+ - V_S = i_A R_A$$

$$i_A = \frac{V_+ - V_1}{R_A}$$

$$V_+ - 0 = i_B R_B$$

But how are you supposed to solve?

Go back & look

Plug in piece eq into current eqs

$$i_A = \frac{V_+ - V_1}{R_A}$$

$$i_B = \frac{V_+ - 0}{R_B}$$

$$\frac{V_+ - V_1}{R_A} - \frac{V_+}{R_B} = 0$$

(28)

Same denom

$$\frac{(V_+ - V_i)R_B - V_+ R_A}{R_A R_B} = 0$$

$$(V_+ - V_i)R_B - V_+ R_A = 0 \quad \text{Petc}$$

$$(V_+ - V_i)R_B = -V_+ R_A$$

$$R_B V_+ - R_B V_i = -V_+ R_A$$

$$R_B V_+ + V_+ R_A = R_B V_i$$

$$V_+ (R_B + R_A) = R_B V_i$$

$$V_+ = \frac{R_B V_i}{R_B + R_A}$$

$$V_i = V_s$$

$$V_+ = V_s \frac{R_B}{R_B + R_A} \quad \text{there we go}$$

I think it is all what you want to target
 But to find V at least, start w/ i current
 and sub in pieces MVCC

I need to have a plan \rightarrow not guess + check