6.01 Final Exam Spring 2011

Name:	Solutions	Section:

These solutions do not apply for the conflict exam.

Enter all answers in the boxes provided.

Clearly written work will be graded for partial credit.

During the exam you may

- refer to any printed materials
- use a calculator

You may not

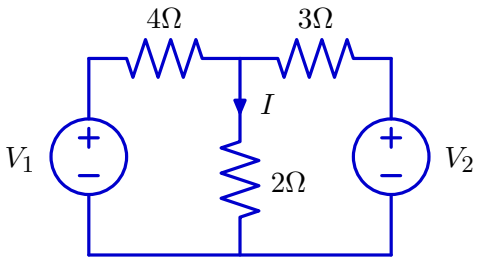
- use a computer, phone, or music player

For staff use:

1.	/8
2.	/8
3.	/6
4.	/18
5.	/11
6.	/18
7.	/15
8.	/16
total:	/100

1 Alternative Excitations (8 points)

Find the relation between V_1 and V_2 that must hold so that $I = 1A$.



Express the relation between V_1 and V_2 as an equation, and enter the equation in the box below.

equation: $3V_1 + 4V_2 = 26$

Let e represent the node where the resistors meet. Applying KCL at that node yields

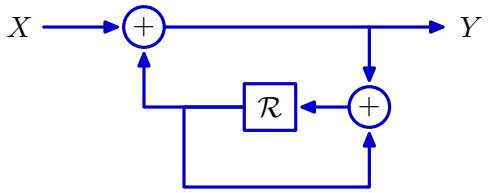
$$\frac{e - V_1}{4} + \frac{e - V_2}{3} + \frac{e}{2} = 0\,.$$

In order for $I = 1A$, e must be 2V. Setting $e = 2V$ in the previous equation yields

$$3V_1 + 4V_2 = 26$$

2 Lots of Feedback (8 points)

Consider the following system.



Part a. Enter the difference equation that relates the input sequence $x[n]$ and output sequence $y[n]$ for this system.

difference equation: $y[n] - 2y[n-1] = x[n] - x[n-1]$

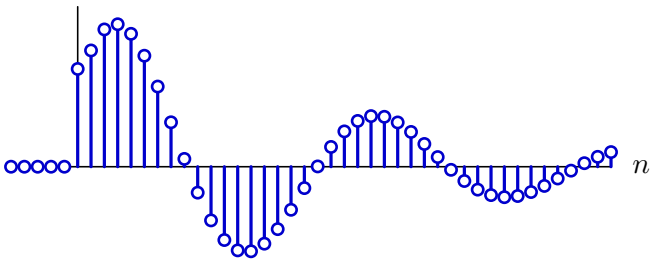
Part b. Enter the pole(s) of the system in the box below. If there are multiple poles, separate them with commas. If there are no poles, enter **none**.

poles: 2

Part c. Assume that you are given a feedback system whose performance is determined by a gain factor K. The following table specifies the two poles that result for six different values of K. The poles are first described by their real and imaginary parts. The same poles are then also described by their magnitude and angles.

K	poles (Cartesion form) real $\pm$ imaginary	poles (polar form) magnitude, angle
0.2	$0.3 \pm 0.1j$	0.316, $\pm$0.322
0.4	$0.6 \pm 0.2j$	0.632, $\pm$0.322
0.6	$0.9 \pm 0.3j$	0.948, $\pm$0.322
0.8	$1.2 \pm 0.4j$	1.264, $\pm$0.322
1	$1.5 \pm 0.5j$	1.581, $\pm$0.322
2	$3.0 \pm 1.0j$	3.162, $\pm$0.322

Which value of K could give rise to the following response?



K = 0.2 or 0.4 or 0.6 or 0.8 or 1 or 2 or **none**: 0.6

The period is approximately 20. Therefore, the angle of the dominant pole must be approximately $2\pi/20 \approx 0.314$, which doesn't eliminate any of the possibilities. The unit sample response oscillates with some decay. Therefore, the magnitude of the dominant pole must be a bit less than 1, limiting the answer to one of the first three cases. The magnitude is bigger than 0.63 since the response is shrinking less than 33% per sample. Thus the answer must be K = 0.6.

3 OOP (6 points)

You are given the following (incomplete) definitions:

```
class MITstudent:
    GIR = ['5.111', '7.011', '8.01', '8.02', '18.01', '18.02',
           'rest1', 'rest2', 'instlab',
           'hass1', 'hass2', 'hass3', 'hass4',
           'hass5', 'hass6', 'hass7', 'hass8']
    DEPT = []
    def __init__(self, AP = []):
        self.taken = AP
    def take(self, courses):
        for course in courses:
            self.taken.append(course)
    def doneReq(self, req):
        missing = []
        for r in req:
            if not r in self.taken:
                print r +' is missing'
                missing.append(r)
        return len(missing) == 0
    def done(self):
        # your code here

class Course62(MITstudent):
    DEPT = ['6.01', '6.02', '18.03', '6.041', '6.002', '6.003', '6.004', '6.005',
            '6.011', '6.013', '6.033', '6.115', '6.813', '6.336',
            '6.AUT', '6.AUP']
```

Note that the definition for each subclass of MITstudent, such as Course62, will define a `DEPT` attribute specifying the departmental requirements.

This code is intended to be used to track the courses taken by a student, and to test whether the student has satisfied all of the GIR and departmental requirements. A sample use is shown below:

```
Chris = Course62(['18.01'])
Chris.take(['5.111', '18.02', '8.01', 'hass1'])
Chris.take(['18.03', '8.02', '6.01', 'hass2'])
Chris.take(['6.02',  '6.041', '7.011', 'hass3'])
Chris.take(['6.002', '6.003', '6.013', 'hass4'])
Chris.take(['6.004', '6.005', '6.011', 'hass5'])
Chris.take(['6.013', '6.115', '6.336', 'hass6'])
print 'Done? ', Chris.done()
Chris.take(['6.AUT', '6.033', '6.813', 'hass7'])
Chris.take(['6.AUP', 'random', 'random', 'hass8'])
print 'Done? ', Chris.done()
```

3.1 Done

Write the `done` method of the `MITstudent` class so that it checks both the GIR and department requirements and returns a boolean (`True` if both requirements are complete; `False` otherwise).

```
def done(self):
    doneGIR = self.doneReq(self.GIR)
    doneDept = self.doneReq(self.DEPT)
    return doneGIR and doneDept
```

3.2 Take

The previous definitions have one fundamental flaw: they don't take into acount that '6.01' and '6.02' together satisfy the Institute Lab requirement 'instlab' under the GIRs.

We can address this flaw by adding a `take` method to the Course62 class to handle this issue. Write the `take` method below; do not duplicate existing code.

```
def take(self, courses):
    MITstudent.take(self, courses)
    if '6.01' in self.taken and '6.02' in self.taken and not 'instlab' in self.taken:
        self.taken.append('instlab')
```

4 Getting from A To B (18 points)

You are building a system to help people plan routes in a city, by a combination of walking and driving. Here is a class definition that characterizes the number of minutes to traverse a link via each of these modes.

```
class Link:
    def __init__(self, walkTime, driveTime):
        self.walkTime = walkTime
        self.driveTime = driveTime
```

Thus `Link(20,10)` specifies a link that takes 20 minutes to walk or 10 minutes to drive. If one of these options is not available (e.g., it is a footpath, or a freeway), then the time can be assigned a very large numerical value (called `far`). For example, `Link(3,far)` specifies a footpath that takes 3 minutes to walk and cannot be driven.

4.1 Personalized cost functions

Different people have different preferences about walking versus driving. One way to account for personal preferences is with costs, as follows. Let `dm` represent the "driving multiplier." Then define the cost of walking to be equal to the number of minutes in the walk, and the cost of driving to be `dm` times the number of minutes in the drive. Thus, if `dm` is 10, the cost of driving for 1 minute is the same as the cost of walking for 10 minutes. Similarly, if `dm` is 0.5, the cost of driving for 1 minute is the same as the cost of walking for 0.5 minutes.

A *cost function* takes two inputs:

- an object of type `Link`, and
- a boolean indicating whether a car is available.

It returns a numeric value, which is the cost of traversing that link. Note that if the car is not available, driving is not a valid option. Assume a person will choose to use whichever available transportation mode has the least cost.

Write a procedure `makeCostFunction(dm)` that takes the driving multiplier for a person and returns a cost function for that person.

```
def makeCostFunction(dm):
    def costFunc(link, car):
        if car:
            return min(link.walkTime, dm*link.driveTime)
        else:
            return link.walkTime
```

4.2 Going with the crowd

Now, assume a whole group of people want to go somewhere together. We would like to select a transportation modality that has the lowest cost averaged over the whole group.

Write a procedure `makeGroupCostFunction(costFunctions)` that takes a list of cost functions (one for each of the people in the crowd) and returns a new cost function for the whole group.

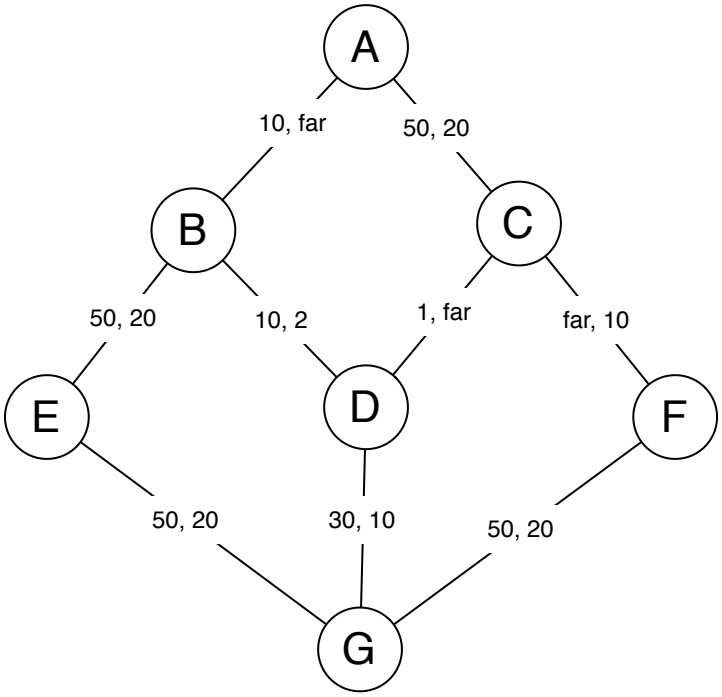
For full credit, use a list comprehension.

You may assume a procedure `average`, which takes a list of numbers as input and returns the average.

```
def makeGroupCostFunction(costFunctions):
    def costFunction(link, car):
        return average([f(link, car) for f in costFunctions])
    return costFunction
```

4.3 Finding a path

Here is an example graph.



Here it is, formalized in Python. It is like the maps we have used before, but instead of a single cost associated with each link, we have an instance of the class `Link`.

```
far = 1000000000
map1 = {'A' : [('B', Link(10, far)),
               ('C', Link(50, 20))],
        'B' : [('A', Link(10, far)),
               ('D', Link(10, 2)),
               ('E', Link(50, 20))],
        'C' : [('A', Link(50, 20)),
               ('D', Link(1, far)),
               ('F', Link(far, 10))],
        'D' : [('B', Link(10, 2)),
               ('C', Link(1, far)),
               ('G', Link(30, 10))],
        'E' : [('B', Link(50, 20)),
               ('G', Link(50, 20))],
        'F' : [('C', Link(far, 10)),
               ('G', Link(50, 20))],
        'G' : [('E', Link(50, 20)),
               ('D', Link(30, 10)),
               ('F', Link(50, 20))]}
```

Assuming that the person always has the option to take a car, what is the best path for a person with `dm = 0.1` from `A` to `G`? What is its cost?

best path = ____ associated cost = ____

4.4 Domain model

Define a class `TransportationSearch`, which is a subclass of `sm.SM`, suitable for use as input to `ucSearch.smSearch`. Its initializer should take a map and a cost function as input. You can just define `self.legalInputs` to be [0, 1, 2, 3]. You do not need to specify the `startState` or the `done` test. Be sure to handle the case when the action is illegal (specifies a choice that does not exist) by remaining in the same state with a cost of 1. You can assume that the person always has the option of taking a car.

```
class TransportationSearch(sm.SM):
 def __init__(self, imap, cf):
       self.map = imap
       self.cf = cf
       self.legalInputs = [0, 1, 2, 3]

   def getNextValues(self, state, a):
       if a >= len(self.map[state]):
           return (state, 1)
       nextState, link = self.map[state][a]
       return (nextState, (nextState, self.cf(link)))
```

4.5 Dude, where's my car?

One problem with our model so far is that it assumes that a person can drive for a segment, walk for a segment, and then drive for another segment. That's a problem, unless you can put your car in your pocket while you walk. So, we really need to stipulate that you can drive for any number of the initial segments of the path, but if you ever walk or take public transportation, you have to park your car, and cannot use it any further on this path.

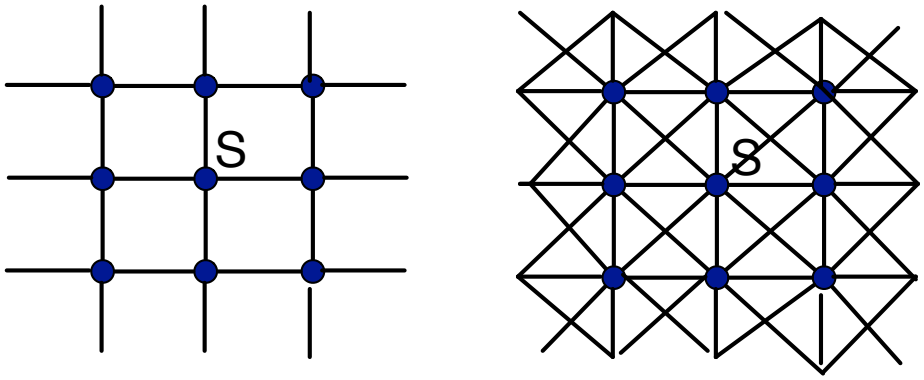
Briefly describe a state and action space for this new version of the problem.

If the person has a car available initially and starts in location `'A'`, what is the initial state?

5 Grid Search (11 points)

Consider a uniform grid that extends indefinitely in all directions. The length between points in the vertical and horizontal directions is 1 meter.

We start out at the grid point (state) marked S; the goal is far away and not shown in the figures.



In the grid on the left, in each state there are four actions that lead to four other states. In the one on the right there are eight actions per state.

In all the questions, assume that the following two pruning rules apply.

"Pruning Rule 1: Do not consider a path that visits the same state twice."

"Pruning Rule 2: If multiple actions lead to the same state, consider just one of them."

Note that a path of length 1 has exactly one action and a path of length 2 has exactly 2 actions.

5.1 Breadth First Search (BF)

- Using BF in the 4-action grid, how many paths of length 1 are ever added to the agenda?

4

- Using BF in the 4-action grid, how many paths of length 2 are ever added to the agenda?

12

- Using BF in the 8-action grid, how many paths of length 1 are ever added to the agenda?

8

- Using BF in the 8-action grid, how many paths of length 2 are ever added to the agenda?

56

5.2 Breadth First Search + Dynamic Programming (BF+DP)

- Using BF+DP in the 4-action grid, how many paths of length 1 are ever added to the agenda? **4**
- Using BF+DP in the 4-action grid, how many paths of length 2 are ever added to the agenda? **8**
- Using BF+DP in the 8-action grid, how many paths of length 1 are ever added to the agenda? **8**
- Using BF+DP in the 8-action grid, how many paths of length 2 are ever added to the agenda? **16**

5.3 Uniform Cost Search (UC)

Assume that the cost of traversing a path is the Euclidean length of the path.

Using UC (NO DP) in the 8-action grid,

- What is the total cost of the second path **expanded**? **1**
- What is the total cost of the sixth path **expanded**? **$\sqrt{2}$**
- What is the total cost of the tenth path **expanded**? **2**

5.4 Heuristics

The "Manhattan distance" between two states (x_1, y_1) and (x_2, y_2) on the grid is:

$$|x_2 - x_1| + |y_2 - y_1|$$

Assume that we are using A* search and we use this distance as an heuristic.

- In the 4-action grid, are we **guaranteed** to find the optimal path?

Yes/No? Yes

Explain briefly.

Manhattan distance is the actual distance on the 4-action grid.

- In the 8-action grid, are we **guaranteed** to find the optimal path?

Yes/No? No

Explain briefly.

Manhattan distance is not an underestimate on the 8-action grid. Diagonal paths are shorter than the Manhattan distance between their endpoints.

6 Funky Elevator (18 points)

You're riding in an elevator in a weird old building. It has three buttons, labeled 'up', 'down', and 'door-close'; and it has a numeric floor indicator, which sometimes shows you a floor number and sometimes is turned off. There are 5 floors in the building. We are interested in estimating what floor the elevator is on, given the history of buttons you have pressed and numbers you have seen on the display. Your interaction with the elevator proceeds as follows:

1. The elevator opens its doors (you may get on or off);
2. The display indicates a floor number, or possibly fails to light up;
3. You push a button; and
4. The elevator moves to a new floor.

6.1 Nominal transitions

You have three possible actions: push the 'up' button, push the 'down' button, or push the 'door-close' button. The elevator was *intended* to work as follows: if you push the 'up' button, then as a result, the elevator travels up one floor (but not above floor 4); if you push the 'down' button, then as a result, the elevator travels down one floor (but not below floor 0); if you push the 'door-close' button, then the elevator travels one floor in the direction specified by the last 'up' or 'down' button that was pushed.

The states of the system are specified by the elevator's floor and the direction that it is traveling.

You are sure you enter the elevator on floor 2, but you think it is equally likely that the elevator is currently going up or going down. Indicate the initial belief state in the following table (states with probability of zero can be left blank).

	0	1	2	3	4
up					
down					

6.2 Noisy transitions

In fact, this elevator is not so reliable. If you push the 'up' button, then if it was already going up, it will go up one floor. If it was going down, then with probability 3/4, it will change directions and go up one floor, otherwise it will go down one floor. Similarly, if you push the 'down' button, then if it was already going down, it will go down one floor; if it was already going up, then with probability 3/4, it will change directions and go down one floor, otherwise it will go up one floor. If you push the 'door-close' button, it will move one floor in the direction it moved on the previous step.

Starting from the initial belief state, what is the belief state of a state estimator after performing action `'door-close'` for 2 steps?

	0	1	2	3	4
up					
down					

Starting from the initial belief state, what is the belief state of a state estimator after performing action `'up'` and then performing action `'door-close'`?

	0	1	2	3	4
up					
down					

Starting from the initial belief state, what is the belief state of a state estimator after performing action `'up'` for two steps?

	0	1	2	3	4
up					
down					

6.3 Adding observations

When the elevator is on some floor f, there is a probability 0.2 that the display will be `None`, a probability 0.4 that the display will be f, probability 0.2 that it will display $f + 1$, and probability 0.2 that it will display $f - 1$. This elevator is so crazy, that it will actually potentially display -1 or 5, so we don't need to worry about special handling for the case of the elevator being on floor 0 or 4.

Starting from the initial belief state, what is the belief state after the observation-action sequence:

```
[(None, 'door-close'), (3, 'door-close')]
```

	0	1	2	3	4
up					
down					

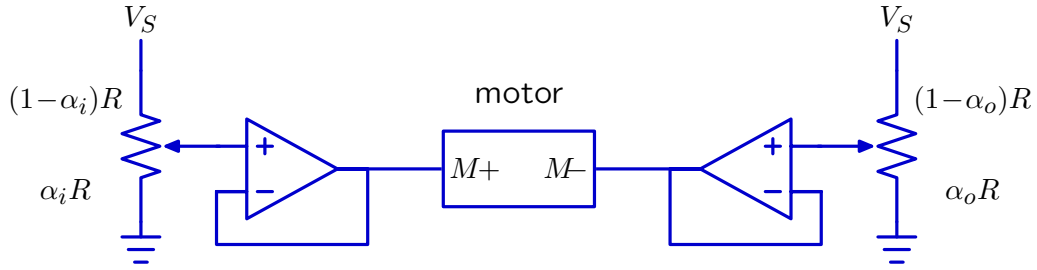
Starting from the initial belief state, what is the belief state after the observation-action sequence:

```
[(3, 'door-close'), (2, 'door-close')]
```

	0	1	2	3	4
up					
down					

7 Position Controller (15 points)

The following figure shows a motor controller. A human can turn the left potentiometer (the *input pot*). Then the motor will turn the right potentiometer (the *output pot*) so that the shaft angle of the output pot tracks that of the input pot.



The dependence of the pot resistances on shaft angle is given in terms of α, which varies from 0 (most counterclockwise position) to 1 (most clockwise position). The resistance of the lower part of the pot is αR and that of the upper part is $(1-\alpha)R$, where $R = 1000\Omega$.

Notice that if $\alpha_i > \alpha_o$, then the voltage to the motor motor ($V_{M+} - V_{M-}$) is positive, and the motor turns clockwise (so as to increase α_o) — i.e., **positive motor voltage** $\rightarrow$ **clockwise rotation**.

Part a. Determine an expression for V_{M+} in terms of α_i, R, and V_S. Enter your expression in the box below.

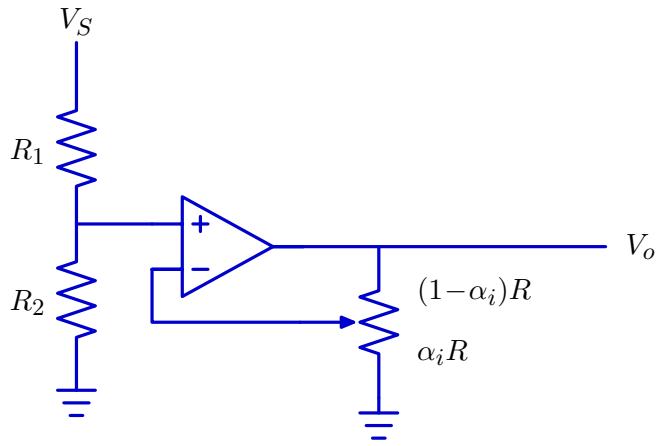
$V_{M+} =$ $\alpha_i V_S$

The output of the voltage divider is

$$V_+ = \frac{\alpha_i R}{\alpha_i R + (1-\alpha_i)R} V_S = \alpha_i V_S \, .$$

The op-amp provides a gain of 1, so $V_{M+} = V_+$.

Part b. The following circuit produces a voltage V_o that depends on the position of the input pot.



Determine an expression for the voltage V_o in terms of α_i, R, R_1, R_2, and V_S. Enter your expression in the box below.

$V_o =$ $\boxed{\dfrac{R_2 V_S}{(R_1 + R_2)\alpha_i}}$

The positive input to the op-amp is connected to a voltage divider with equal resistors so

$$V_+ = \frac{R_2}{R_1 + R_2} V_S \, .$$

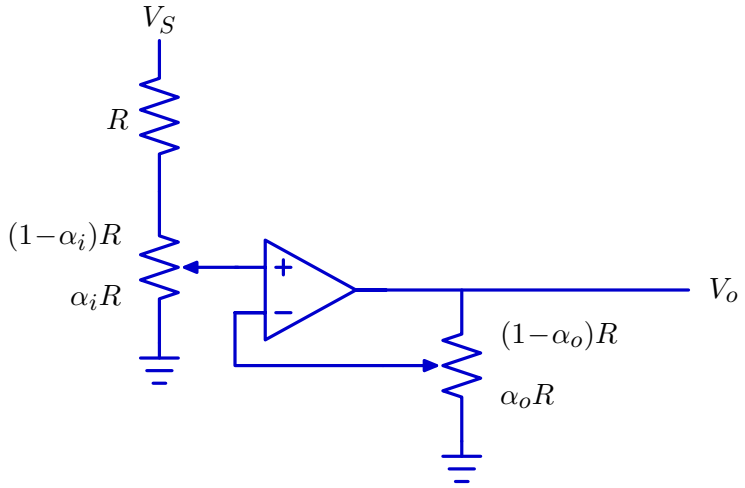
The input pot is on the output of the op-amp, so

$$V_- = \frac{\alpha_i R}{\alpha_i R + (1 - \alpha_i) R} V_o = \alpha_i V_o \, .$$

In an ideal op-amp, $V_+ = V_-$ so

$$V_o = \frac{R_2 V_S}{(R_1 + R_2)\alpha_i} \, .$$

Part c. The following circuit produces a voltage V_o that depends on the positions of both pots.



Determine an expression for V_o in terms of α_i, α_o, R, and V_S. Enter your expression in the box below.

$V_o =$ $\frac{\alpha_i}{\alpha_o}\frac{V_S}{2}$

The positive input to the op-amp is connected to pot 1 so that

$$V_+ = \frac{\alpha_i R}{\alpha_i R + (1-\alpha_i)R + R} V_S = \frac{\alpha_i V_S}{2} .$$

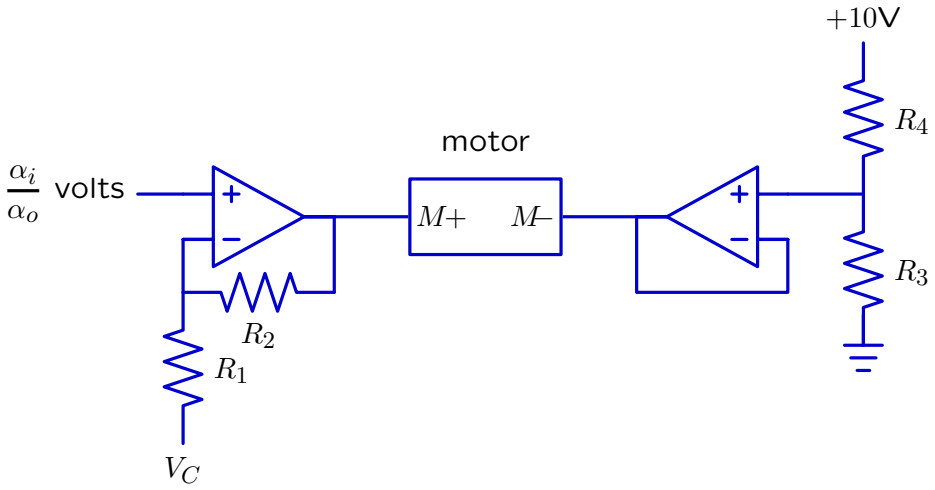
The output pot is on the output of the op-amp, so

$$V_- = \frac{\alpha_o R}{\alpha_o R + (1-\alpha_o)R} V_o = \alpha_o V_o .$$

In an ideal op-amp, $V_+ = V_-$ so

$$V_o = \frac{\alpha_i}{\alpha_o}\frac{V_S}{2} .$$

Assume that we are provided with a circuit whose output is $\frac{\alpha_i}{\alpha_o}$ volts. We wish to determine if it is possible to design a motor controller of the following form



so that the motorshaft angle (which is proportional to α_o) will track the input pot angle (which is proportional to α_i).

Part d. Assume that $R_1 = R_3 = R_4 = 1000\Omega$ and $V_C = 0$. Is it possible to choose R_2 so that α_o tracks α_i?

Yes or **No**: Yes

If **yes**, enter an acceptable value (a number) for R_2.

$R_2 =$ 4000Ω

If **no**, briefly explain why.

If $R_3 = R_4$ then the right motor input is 5V. If $\alpha_i = \alpha_o$ then the gain of the left op-amp circuit must be 5 so that the motor voltage is 0. The gain is $R_1 + R_2/R_1$, so R_2 must be 4000Ω.

Part e. Assume that $R_1 = R_3 = R_4 = 1000\Omega$ and $V_C = 5V$. Is it possible to choose R_2 so that α_o tracks α_i?

Yes or **No**: No

If **yes**, enter an acceptable value (a number) for R_2.

$R_2 =$

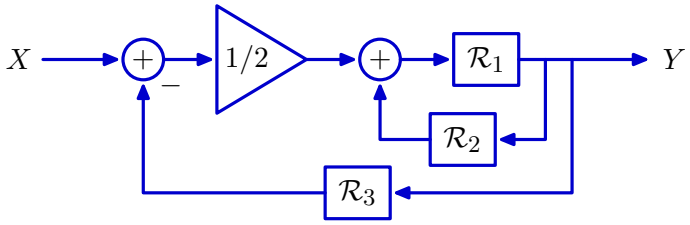
If **no**, briefly explain why.

If $R_3 = R_4$ then the right motor input is 5V. If $\alpha_i = \alpha_o$ then $V_+ = V_- = 1$ for the right op-amp. We need the left motor input to be 5V. But if the left motor input is 5V and $V_C = 5V$ then V_- must also be 5V, which leads to a contradiction.

8 Delay Removal (16 points)

Your company is interested in ensuring that the system shown below converges to its final value as soon as possible. It has enough money to engineer one of the delays out of the system. Your job is to determine which delay, if removed, would result in the fastest convergence.

Here is the "original system." (It is also shown on the next page, for you to use as a worksheet).



8.1 Analysis

Let H_1 represent the system that results when $\mathcal{R}_1$ is a wire, $\mathcal{R}_2$ is a delay, and $\mathcal{R}_3$ is a delay.

Let H_2 represent the system that results when $\mathcal{R}_2$ is a wire, $\mathcal{R}_1$ is a delay, and $\mathcal{R}_3$ is a delay.

Let H_3 represent the system that results when $\mathcal{R}_3$ is a wire, $\mathcal{R}_1$ is a delay, and $\mathcal{R}_2$ is a delay.

Fill in the following table with properties of H_1, H_2, and H_3.

system function		pole(s)	converge?	oscillate?
$H_1 =$	$\frac{1}{2-\mathcal{R}}$	$\frac{1}{2}$	yes	no
$H_2 =$	$\frac{\mathcal{R}}{2-2\mathcal{R}+\mathcal{R}^2}$	$\frac{1}{2} \pm j\frac{1}{2}$	yes	yes
$H_3 =$	$\frac{\mathcal{R}}{2+\mathcal{R}-2\mathcal{R}^2}$	$-\frac{1}{4} \pm \sqrt{\frac{17}{16}}$	no	no

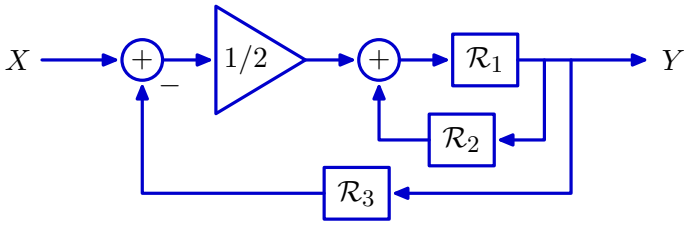
8.2 Recommendation

Which delay should your company remove?

1 or **2** or **3** or **none**: 1

Briefly explain.

Fastest convergence results when the magnitude of the dominant pole is smallest. The magnitudes of the dominant poles for the H_1, H_2, and H_3 systems are $\frac{1}{2}$, $\frac{\sqrt{2}}{2}$, and $\sqrt{\frac{21}{16}}$, respectively. The smallest magnitude occurs for H_1.


X
+
−
1/2
+
$\mathcal{R}_1$
Y
$\mathcal{R}_2$
$\mathcal{R}_3$

Worksheet (intentionally blank)

6.01SC Introduction to Electrical Engineering and Computer Science

Spring 2011