6.01 Final Exam: Spring 2009

Name:

Enter all answers in the boxes provided.
You may use any paper materials you have brought.
No computers, cell phones, or music players.

For staff use:

1.	/14
2.	/14
3.	/14
4.	/14
5.	/15
6.	/14
7.	/15
total:	/100

worksheet

1 Programming (14 points)

In the Equation class, we represent a linear equation

$$a_0x_0 + ... + a_nx_n = c$$

with a list of coefficients, `coeffs`, corresponding to $[a_0, ..., a_n]$ and the `constant`, c.

```
class Equation:
    def __init__(self, coeffs, variableNames, constant):
        self.variableNames = variableNames    # List of variable names
        self.coeffs = coeffs                  # List of coefficients
        self.constant = constant              # Constant (right hand side)
```

Suppose that we had a list of values of `[v0, ..., vn]` for the variables (x_0, ... x_n) in which one of the values is `None` and all of the rest of the values are numbers.

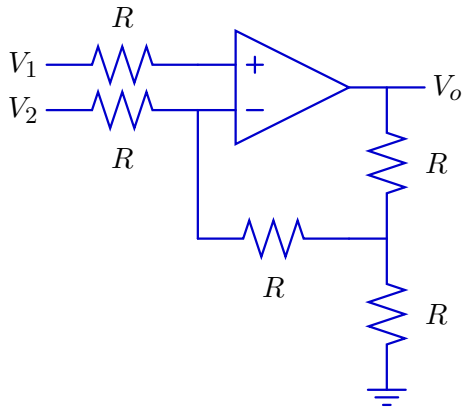
These numeric values are enough to determine a value for the unspecified variable.

Write a procedure `determineValue` that takes an instance of the `Equation` class and a list of values (exactly one of which is `None`) and returns the value of the unspecified variable.

```
def determineValue(eq, values):
```

2 Circuits (14 points)

Let V_- represent the voltage at the negative input of the op-amp in the following circuit.



Notice that if we knew the values of V_o and V_2 then we could calculate the value of V_-. The resulting value of V_- will be a linear combination of V_2 and V_o of the following form:

$$V_- = \alpha V_o + \beta V_2. \tag{1}$$

Part a. Use **Equation 1** to find an expression for the output voltage V_o in terms of the input voltages V_1 and V_2. Which of the following expressions gives the result?

Hint: it is not necessary to determine the values of α and β to do this part of the question.

$V_A = \alpha V_2 - \beta V_1$ $\quad$ $V_B = \dfrac{\alpha V_2 - V_1}{\beta}$ $\quad$ $V_C = \beta V_2 - \alpha V_1$

$V_D = \dfrac{\beta V_2 - V_1}{\alpha}$ $\quad$ $V_E = \dfrac{V_2 - \alpha V_1}{\beta}$ $\quad$ $V_F = \dfrac{V_2 - \beta V_1}{\alpha}$

$V_G = \alpha V_1 - \beta V_2$ $\quad$ $V_H = \dfrac{\alpha V_1 - V_2}{\beta}$ $\quad$ $V_I = \beta V_1 - \alpha V_2$

$V_J = \dfrac{\beta V_1 - V_2}{\alpha}$ $\quad$ $V_K = \dfrac{V_1 - \alpha V_2}{\beta}$ $\quad$ $V_L = \dfrac{V_1 - \beta V_2}{\alpha}$

$V_o =$ (V_A or V_B or ... or V_L or `none`)

Part b. Find α and β in **Equation 1** (previous page). **Hint:** Try superposition.

$\alpha =$ $\beta =$

Part c. Combine the results of parts a and b to determine an expression for V_o in terms of V_1 and V_2.

$V_o =$

3 OOP (14 points)

We wish to implement a set of classes to model graphs of the type that we used to describe search problems.

Graphs have **nodes** and **edges**. A directed edge indicates a connection from the first node to the second. An undirected edge can be thought of as two directed edges, one from the first node to the second and another one going the other way. Nodes will be identified with names that are represented using Python strings.

Implement a base class called `DirectedGraph`, which has the following methods:

- `hasNode` - given a node, returns `True` when the node exists in the graph and false otherwise.
- `addNode` - given a node, if it's not already present, it adds it to the graph, initially with no edges.
- `addEdge` - given two nodes, add a connection from the first to the second. If either node is not already present, it is added to the graph.
- `children` - given a node, returns a list of nodes that can be reached from that node by traversing a single arc. If the node is not present, it returns an empty list.

Here is an example use:

```
>>> g = DirectedGraph()
>>> g.addNode('A')
>>> g.addEdge('A','B')
>>> g.children('A')
['B']
>>> g.children('B')
[]
```

Think very carefully about:

- How you will keep track of the nodes in the graph.
- How you will keep track of which nodes are connected.

Each of the methods should be short.

Define this class and its methods.

```
class DirectedGraph:
```

Define a class `UnDirectedGraph` which has all the same methods as `DirectedGraph`, except that `addEdge`, adds edges in both directions. Here is an example use:

```
>>> g = UnDirectedGraph()
>>> g.addEdge('A','B')
>>> g.children('A')
['B']
>>> g.children('B')
['A']
```

Define this class. Use inheritance to avoid rewriting all of the methods.

4 Graph Search (14 points)

We can use the `DirectedGraph` class (previous problem) to define a search problem. We will define a state machine class, `GraphSM`, whose states correspond to the nodes in the graph and whose state transitions correspond to the edges in the graph.

The input to the machine is an integer indicating which of the children of the current node will become the next state. If the input is greater than or equal to the length of the list of children, then the state remains unchanged.

A `GraphSM` machine is initialized with an instance of `DirectedGraph`, a start node and an integer indicating the maximum number of children of any node in the graph.

```
>>> g = DirectedGraph()
>>> for s in ['S', 'A', 'B', 'C', 'D', 'E', 'F', 'G']:
        g.addNode(s)
>>> for (a, b) in [('S', 'A'), ('S', 'B'), ('A', 'C'), ('B', 'C'), ('B', 'D'),
                   ('C', 'E'), ('D', 'E'), ('D', 'F'), ('F', 'G')]:
        g.addEdge(a,b)
>>> m = GraphSM(g, 'S', 2)
>>> m.transduce([1,1,1,0])
['B', 'D', 'F', 'G']
>>> search.smSearch(GraphSM(g, 'S', 2), goalTest=lambda s: s == 'E')
[(None, 'S'), (0, 'A'), (0, 'C'), (0, 'E')]
```

4.1 GraphSM

Define the `GraphSM` class. Define all the methods and attributes that you need for the state machine to be used for a search, as shown in the example above. You do not need to define a `done` method.

```
class GraphSM(SM):
    def __init__(self, graph, startNode, maxChildren):
```

4.2 Search

For the graph defined by the code shown above:

- Show the sequence of partial paths **expanded** by depth-first search **without** DP from S to G. **Assume children of a state are pushed on the agenda (visited) in reverse alphabetical order**. There are more than enough slots.

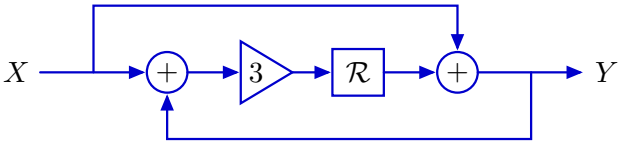
step	partial path expanded
1	S
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	

- List in order the states that are **visited** by breadth-first search **without** DP starting at S and going to G. If a state is visited more than once, include it each time it is visited.

- List in order the states that are **visited** by breadth-first search **with** DP starting at S and going to G. If a state is visited more than once, include it each time it is visited.

5 System Functions (15 points)

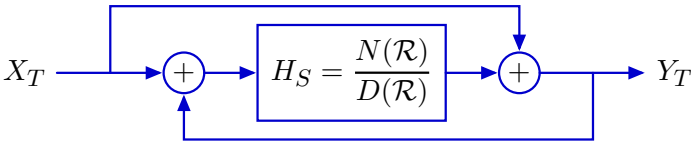
Consider the following system.



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

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

Part c. Consider a generalization of the previous system with the following form.



Write a Python function called `YinYang` that takes a single input parameter `HS`, which is a `SystemFunction` that represents H_S, and returns a `SystemFunction` that represents $H_T = Y_T(\mathcal{R})/X_T(\mathcal{R})$.

```
def YinYang(HS):
```

Part d. Consider the following code, which builds on the `YinYang` function in part c.

```
def TwoYinYangs(K):
    g1 = sf.Gain(K)
    h1 = YinYang(sf.R())
    h2 = YinYang(sf.R())
    return sf.FeedbackSubtract(sf.Cascade(g1,sf.Cascade(h1,h2)))
```

Enter the system function that will be returned by `TwoYinYangs(3)` in the box below. Express your answer as a ratio of polynomials in $\mathcal{R}$.

$H(\mathcal{R}) =$

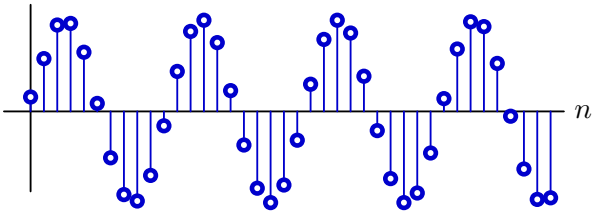
Part e. Executing the following code

```
for K in [0.01, 0.02, 0.05, 0.1, 0.2, 0.5]:
    print K
    for p in TwoYinYangs(K).poles():
        print ">> x,y= ",p,"    r,theta= ",sf.complexPolar(p)
```

produces the following output.

```
0.01
>> x,y=  (0.980+0.198j)     r,theta=  (1.0,  0.199)
>> x,y=  (0.980-0.198j)     r,theta=  (1.0, -0.199)
0.02
>> x,y=  (0.961+0.277j)     r,theta=  (1.0,  0.281)
>> x,y=  (0.961-0.277j)     r,theta=  (1.0, -0.281)
0.05
>> x,y=  (0.905+0.426j)     r,theta=  (1.0,  0.440)
>> x,y=  (0.905-0.426j)     r,theta=  (1.0, -0.440)
0.1
>> x,y=  (0.818+0.575j)     r,theta=  (1.0,  0.613)
>> x,y=  (0.818-0.575j)     r,theta=  (1.0, -0.613)
0.2
>> x,y=  (0.667+0.745j)     r,theta=  (1.0,  0.841)
>> x,y=  (0.667-0.745j)     r,theta=  (1.0, -0.841)
0.5
>> x,y=  (0.333+0.943j)     r,theta=  (1.0 , 1.231)
>> x,y=  (0.333-0.943j)     r,theta=  (1.0, -1.231)
```

Which value of K would give rise to the following unit sample response?



K =? (0.01 or 0.02 or 0.05 or 0.1 or 0.2 or 0.5 or **none**) :

6 State Estimation (14 points)

We are interested in estimating and predicting the amount of traffic on a particular segment of road. We will model the traffic level as being either Low, Medium, or High.

Observations

When a specially-equipped car drives down the road of interest, we can measure its speed. Knowing the car's speed gives us information about the traffic on the road. The **conditional probability distribution** $\Pr(\mathit{Speed} \mid \mathit{Traffic})$ is specified in the following table.

	Speed							
Traffic	<10	10-20	20-30	30-40	40-50	50-60	60-70	70-80
Low	0	0.1	0.1	0.1	0.2	0.2	0.2	0.1
Med	0.2	0.2	0.2	0.2	0.2	0	0	0
High	0.5	0.5	0	0	0	0	0	0

Part a. Imagine that our initial belief about the traffic is:

$$\Pr(S_0 = \mathit{Low}) = 0.5, \qquad \Pr(S_0 = \mathit{Med}) = 0.25, \qquad \Pr(S_0 = \mathit{High}) = 0.25.$$

Now, we make a single observation O_0 of the speed of a car on the road.
In the following, we consider three possible outcomes: $O_o = 75$ or $O_o = 45$ or $O_o = 5$.

- What would be our updated belief given that the single observation is $O_o = 75$?

$\Pr(S_0 = \mathit{Low} \mid O_0 = 75) =$ ☐ $\Pr(S_0 = \mathit{Med} \mid O_0 = 75) =$ ☐

$\Pr(S_0 = \mathit{High} \mid O_0 = 75) =$ ☐

- What would be our updated belief given that the single observation is $O_o = 45$?

$\Pr(S_0 = \mathit{Low} \mid O_0 = 45) =$ ☐ $\Pr(S_0 = \mathit{Med} \mid O_0 = 45) =$ ☐

$\Pr(S_0 = \mathit{High} \mid O_0 = 45) =$ ☐

- What would be our updated belief given that the single observation is $O_o = 5$?

$\Pr(S_0 = \mathit{Low} \mid O_0 = 5) =$ ☐ $\Pr(S_0 = \mathit{Med} \mid O_0 = 5) =$ ☐

$\Pr(S_0 = \mathit{High} \mid O_0 = 5) =$ ☐

Transitions

Now, consider how the traffic state changes over time. We'll think about the traffic system as having two possible inputs (actions), N, and A. Input $I_t = N$ means that there has been no disturbance on that stretch of road on step t; input $I_t = A$ means that there has been an accident on that stretch of road at step t. Depending on whether there has or has not been an accident, the transition probabilities differ.

The **conditional probability distribution** $Pr(S_{t+1} \mid S_t, I_t = N)$ is given by:

		S_t+1		
	Low	Med	High
	Low	0.9	0.1	0
S_t	Med	0.1	0.8	0.1
	High	0	0.1	0.9

The **conditional probability distribution** $Pr(S_{t+1} \mid S_t, I_t = A)$ is given by:

		S_t+1		
	Low	Med	High
	Low	0	0.1	0.9
S_t	Med	0	0	1
	High	0	0	1

Part b1. If we are certain that the traffic is low at time 0 (that is, $Pr(S_0 = \mathit{Low}) = 1$), and there are no accidents for the next two steps (and we make no observations), what is our belief about the state of the traffic at time 2? Specify the following values:

$Pr(S_2 = \mathit{Low} \mid I_0 = N, I_1 = N) =$

$Pr(S_2 = \mathit{Med} \mid I_0 = N, I_1 = N) =$

$Pr(S_2 = \mathit{High} \mid I_0 = N, I_1 = N) =$

Part b2. If we are certain that the traffic is low at time 0, and there are accidents for the next two steps (and we make no observations), what is our belief about the state of the traffic at time 2? Specify the following values:

$Pr(S_2 = \mathit{Low} \mid I_0 = A, I_1 = A) =$

$Pr(S_2 = \mathit{Med} \mid I_0 = A, I_1 = A) =$

$Pr(S_2 = \mathit{High} \mid I_0 = A, I_1 = A) =$

Combining observation and transition

Part c. Assuming that you initially believe that each traffic state is equally likely, which of the following sequences are possible?

$O_0 = 75, I_0 = A, O_1 = 75, I_1 = N$

possible? (yes/no): If no, briefly explain why (one sentence only).

$O_0 = 75, I_0 = A, O_1 = 15, I_1 = N$

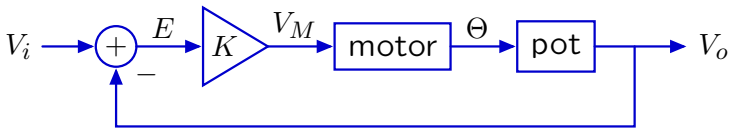
possible? (yes/no): If no, briefly explain why (one sentence only).

$O_0 = 75, I_0 = N, O_1 = 15, I_1 = N$

possible? (yes/no): If no, briefly explain why (one sentence only).

7 Modeling (15 points)

The following block diagram illustrates the signal flow paths through a circuit that is used to control the shaft angle of a motor.



The error E between the input voltage V_i and output voltage V_o is multiplied by the gain K to generate the motor input voltage V_M. Assume that the angular speed of the motor shaft (Ω, not shown) is proportional to V_M so that $\Omega = K_M V_M$. The motor turns the shaft of a potentiometer ("pot") to angle Θ and the output voltage from the potentiometer (V_o) is proportional to the shaft angle, i.e., $V_o = \alpha\Theta$.

We wish to make a model of this system in which the shaft angle Θ at time n is equal to the shaft angle at time $(n-1)$ plus the product of the time between steps, T, times the angular speed Ω at time $(n-1)$,

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

where $\theta[n]$ is the n^{th} sample of the Θ signal and $\omega[n]$ is the n^{th} sample of the Ω signal.

Part a. Draw a block diagram for the **entire** motor system that consists of just adders, gains, and delays. Label the nodes that correspond to V_i, V_o, E, V_M, Ω, and Θ.

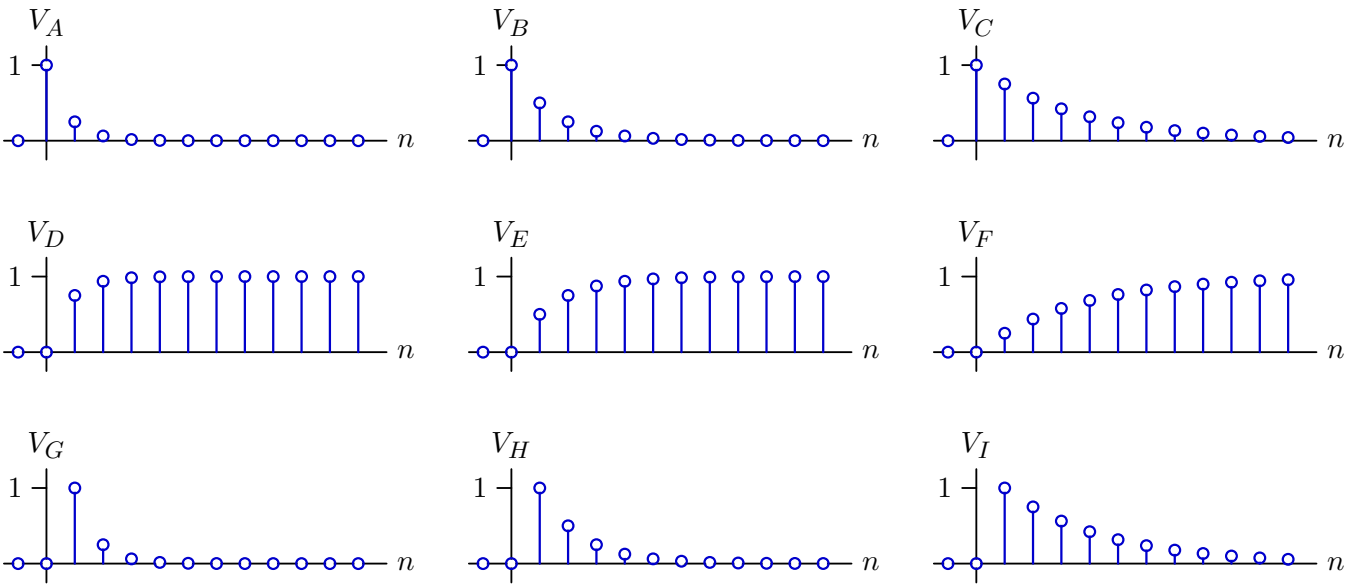
Part b. Determine the system function $H = \dfrac{V_o}{V_i}$. Express your answer as a ratio of two polynomials in $\mathcal{R}$.

$$H = \frac{V_o}{V_i} =$$

Part c. The step response of the system was measured by stepping the input voltage from 0 V for times $n < 0$ to 1 V for times $n \geqslant 0$. Based on this measurement, it was concluded that the system function can be written in the following form:

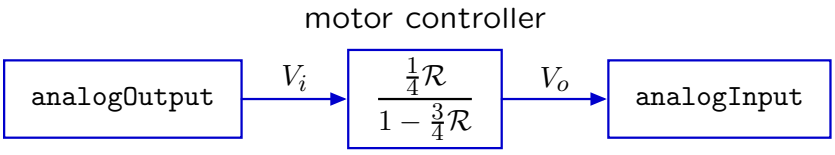
$$H = \frac{V_o}{V_i} = \frac{\frac{1}{4}\mathcal{R}}{1 - \frac{3}{4}\mathcal{R}}$$

Which of the following **step responses** is consistent with the previous equation?



step response = V_A or V_B or ... or V_I or `none`:

Part d. The motor control circuit was wired up to the robot so that the robot's "brain" could "command" the motor input voltage V_i from the `analogOutput` and "read" the resulting motor output voltage V_o via an `analogInput`.



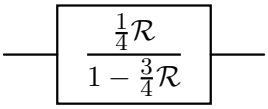
The following code implements a brain that toggles the motor position back and forth by toggling the `analogOutput` between 0.3 V and 0.7 V.

```
class motorControlBrain(sm.SM):
    startState = 0
    def getNextValues(self, state, inp):
        vo = inp.analogInput[0]   # vo is the output voltage from the previous step
        newState = (state+1)%100
        if newState>50:
            vx = 0.7
        else:
            vx = 0.3
        return (newState,io.Action(voltage=vx))
```

A brilliant student figured out that it was possible to speed up the response of the motor controller by changing the control **code** without changing the hardware in any way! The student simply changed the last line of `motorControlBrain` to the following (leaving the other lines as they were):

```
        return (newState,io.Action(voltage=C*vx-D*vo))
```

where `C` and `D` are constants. We will refer to the resulting system as the "CD Motor Controller." Draw a block diagram that relates the input V_X to the output V_o for the CD Motor Controller. Your block diagram should include the original motor controller, which is already provided below.



Part e. Determine the system function $H_X = \dfrac{V_o}{V_X}$ for the CD Motor Controller.

$$H_X = \frac{V_o}{V_X} =$$

Part f. To use the CD Motor Controller, our brilliant student had to determine values of the constants `C` and `D`. The values that were chosen gave rise to the following system function:

$$H_X = \frac{V_o}{V_X} = \frac{\frac{25}{64}\mathcal{R}}{1 - \frac{3}{4}\mathcal{R} + \frac{9}{64}\mathcal{R}^2} .$$

Is the performance of this system better or worse than that of the original motor controller in part c? Briefly explain your reasoning.

worksheet

worksheet

6.01SC Introduction to Electrical Engineering and Computer Science

Spring 2011