

Midterm 1 Practice

This is a collection of problems from past exams. Note that the first two were from the Spring 08 midterm, which was a three-hour take-home. This is much, much longer than our actual exam will be. Problems 6 through 10 constitute the corresponding exam from Spring 09.

↑ skip to that

1 Robot behavior

In the following, we would like our robot behavior to transduce a stream of sensor inputs (sonar readings and poses, as before), to a stream of `io.Action` instances, specifying the forward and angular velocities for the robot.

So, here is a brain that would use a behavior that you write in the following problem:

```
def setup():
    robot.behavior = yourSMHere
    robot.behavior.start()
```

```
def step():
    robot.behavior.step(io.SensorInput()).execute()
```

Since open book would go through + find how I wrote

- (10 points)** Define a class `TurnSmoothly`, which is a subclass of `SM`. It should use a proportional controller to rotate until its odometry heading is `thetaDesired`.
- (10 points)** Build a linear difference equation model of the robot's controller and of the world. Assume that there is one unit of delay in the dependence between the robot's angular velocity and its pose angle, and that there is one unit of delay in setting the desired angular velocity. You may do this algebraically by hand, or by using the `SystemFunction` software, but either way, your answer should be a difference equation, and you should show how you found it.
- (10 points)** Find a gain or gains for your controller that cause the system in your model above to converge monotonically to a heading of `thetaDesired`. Assume that T (the length of a delay) is 0.2. Explain how you found the gain. (You may do this algebraically by hand, or by using the `SystemFunction` software.)
- (5 points)** What is the effect on the stability of your controller of increasing the value of T . Explain your answer. Does it match what you might expect intuitively?
- (7 points)** *This problem may take more time than the others; leave it until the end.*

-I shall print all my code

Now, imagine that instead of giving angular velocity commands to the robot, we can only give angular *acceleration* commands. We'll consider controllers for this problem that are analogous to the "proportionalPlusDelay" controller (with gains K_1 and K_2).

Define a function that creates a `SystemFunction` model of the robot's controller and of the world. Assume that there is one unit of delay in the dependence between the robot's angular velocity and its pose angle, and a one unit of delay in the dependence between the robot's angular acceleration and its angular velocity.

Use Python to search the range of values of each K between -10 and 10. If $K_2 = 0$, are there values of K_1 for which the controller is stable? If $K_1 = 0$, are there values of K_2 for which the controller is stable? Are there combined values of K_1 and K_2 that make the controller stable. In each case, indicate the best gains that you found.

2 Weighted objective

2.1 Preliminary material

The following material was in a tutor problem from last term: A priority queue is a kind of data structure that you can add elements to in any order, but which gives you an operation that allows you to take out the smallest one. We'll assume that the elements being added are numbers, and the ordering is less than.

Create a class, called `PriorityQueue` that implements the priority queue abstract data type. It should have three methods: `__init__`, which initializes the priority queue, `insert`, which takes a number and adds it to the priority queue, and `extract`, which takes the smallest element out of the priority queue and returns it.

There are all kinds of fancy algorithms for doing these operations efficiently; you need not use them here. For example, you might just store the elements in a list. You can use the Python list operations `append`, `remove`, and `min`. Assuming that `items` is a list,

- `items.append(x)` adds element x to the end of the list, and returns `None`
- `items.remove(x)` removes the first occurrence of element x in the list, and returns `None`; it generates an error if x isn't in the list.
- `min(items)` returns the smallest element in the list

There can be multiple copies of a single value in the PQ. In such cases, you should remove each one separately. So, for instance, here's a test interaction your object should support:

```
> pq = PriorityQueue()
> pq.insert(10)
> pq.insert(8)
> pq.insert(10)
> pq.insert(20)
> pq.extract()
8
> pq.extract()
10
> pq.extract()
10
> pq.insert(30)
> pq.extract()
20
```

2.2 Main questions

In this problem, we are going to make a data structure similar to a priority queue (which we saw in tutor problem described above), but with a special method for selecting the element to be removed. We'll call it a WOPQ (weighted-objective priority queue). We can think of it as an abstract data type with the following operations:

- Make a new WOPQ, and store *features*, which is a list of functions, each of which takes an element as input and returns a number, and *weights*, which is a list of positive numbers that sum to 1, of the same length as *features*
- Insert an element to the WOPQ
- Extract the *best* element from the WOPQ and return it after deleting it from WOPQ. An element x is the best element if its priority

$$p(x) = \sum_i f_i(x) \cdot w_i ,$$

where f_i is the i th function in *features* and w_i is the i th weight in *weights*, is greater than or equal to the priorities of all other elements in the WOPQ

A WOPQ shouldn't assume anything about the elements it will hold.

Here is a concrete example of a WOPQ:

- The elements will be lists of numbers
- The features are `sum` and `len`
- The weights are 0.6 and 0.4

In this case, if $a = [1, 2, 3]$, then `sum(a) = 6` and `len(a) = 3`. So its priority $p(a) = 0.6 * 6 + 0.4 * 3 = 4.8$. For another list, $b = [1, 1, 1, 1, 1]$, then `sum(b) = 5` and `len(b) = 5`. So its priority $p(b) = 0.6 * 5 + 0.4 * 5 = 5$. If our WOPQ contained both a and b , and we did an extract operation, then element b would be deleted and returned.

But in other instances of WOPQs, the elements might be system functions and the features related to the poles, or the elements might be records describing bank accounts and the features related to balances and interest rates.

Here is the skeleton of a WOPQ class:

```
class WOPQ:
    def __init__(self, features, weights):
        self.features = features
        self.weights = weights
        self.elements = []

    def insert(self, x):
        self.elements.append(x)
```

```
def extract(self):
    # your code here
```

- (10 points)** Write a Python procedure `argmax` that takes two arguments: the first is a list of elements, and the second is a function from elements to numerical values. It should return the element of the list with the highest numerical score.
- (5 points)** Let's make a WOPQ with fish. Here's the `Fish` class, and some fishy instances:

```
class Fish:
    def __init__(self, length, width):
        self.fishLength = length
        self.fishWidth = width

    def length(self):
        return self.fishLength
    def width(self):
        return self.fishWidth
```

```
Lee = Fish(12, 200)
Evelyn = Fish(30, 40)
Sydney = Fish(60, 23)
```

Provide a Python statement that would construct a WOPQ that measures fish so that the priority of a fish is 0.9 times its length plus 0.1 times its width.

- (10 points)** Write the `extract` method of the WOPQ class, which finds the element with the highest score, removes it from the list of elements, and returns it. Use `argmax`. Remember that if `x` is a list, then `x.remove(5)` will remove the first occurrence of 5 from `x` (but it doesn't return a value).
- (5 points)** Letting `wopq` be the WOPQ you constructed in the previous question, imagine that the following operations have been conducted on it:

```
wopq.insert(Lee)
wopq.insert(Evelyn)
wopq.insert(Sydney)
```

Now, what would be the result of executing

```
wopq.extract()
```

Explain which element was selected for extraction and why.

- (10 points)** After all that work, you discover that someone has already written a FPQ class, whose initialization method takes a function `priorityFun` as an argument, and whenever the `extract` method is called, it extracts the item with the highest value of that function.

That is, we have:

```
class FPQ:
    def __init__(self, priorityFun):
        self.priorityFun = priorityFun
        <etc>
```

Provide an implementation of the WOPQ class that uses inheritance to take advantage of the existing FPQ class.

6. (10 points) *This problem may take more time than the others; leave it until the end.*

Now, suppose you are given a list of candidate features, which are, as above, procedures that map an element into a number. We'd like to select a subset of these features, to use later in our WOPQ. Write a procedure `selectFeatures` that takes as input

- a list of candidate features
- a number n (which less than the number of features in the list of candidate features)
- a list of elements in our domain (to which the features may be applied)

Define the *range* of a feature f on the list of elements E to be $\max_{x \in E} f(x) - \min_{x \in E} f(x)$. (That is, the maximum value of the feature on all of the examples minus the minimum value of the feature on all of the examples). Your procedure should return a list of n of the given features for which the range on the given set of elements is the largest.

Note that, to sort a list of items according to some function of their values, you can use the `sorted` function. The example below sorts a list of numbers according to the squares of their values:

```
>>> foo = [1, 2, -30, -20, 3, 90, -27]
>>> sorted(foo, key = lambda x: x * x)
[1, 2, 3, -20, -27, -30, 90]
```

You can put any function of a single argument in as the key parameter to `sorted`. It is most straightforward to use and understand if the values output by the key function are numbers.

3 More OOP

You are told that we have a class A, with one initialization argument, with a method f that takes one argument and returns a number.

```
>>> x = A(2)
>>> x.f(3)
9
```

Define a class B that is a subclass of class A. Class B has the same initialization as A and a method f that returns twice whatever value the f method of A would return. You don't need to know what the code for A is.

```
>>> x = B(2)
>>> x.f(3)
18
```

```
def B(A)
def f(inp)
return A.f(inp) * 2
```

4 In the Tank (6 points)

Imagine a cylindrical water tank with a source of water entering the bottom of the tank and a sensor to measure the depth of the water. The source of water can be adjusted so that water enters (or exits) the tank at a desired rate. In addition to this controlled source of water, there is also a leak in the tank, through which water flows regardless of the controlled flow. The diameter of the tank is such that one inch in depth is one gallon of water. The following variables characterize this problem:

- $d[t]$ is the depth, in inches, of the water in the tank at time t .
- $v[t]$ is the volume of water that flows through the controlled source of water in the time interval $[t, t + 1]$.
- $o[t]$ is the volume of water that leaks from the tank in the time interval $[t, t + 1]$. Assume $o[t]$ is proportional to $d[t]$, with a proportionality constant of 0.1
- $g[t]$ is the desired (goal) depth of water (in inches) in the tank at time t
- $s[t]$ is the sensed depth (in inches) of water in the tank at time t ; assume the sensor introduces one unit of delay, so that $s[t] = d[t - 1]$

they are having you write each part

a. Write a difference equation that describes the depth of water in the tank ($d[\cdot]$) in terms of the flow through the controlled source ($v[\cdot]$).

b. Write a difference equation describing a proportional controller that specifies $v[\cdot]$ as a function of sensed depth $s[\cdot]$ and desired depth $g[\cdot]$. Let α represent the constant of proportionality.

$$d[n] = v[n-1] + d[n-1]$$

$$d[t+1] = d[t] + v[t] - o[t]$$

$$d[n-1] + v[n-1] - o[n-1]$$

$$- 0.1 d[n-1]$$

*and if using one, $d[t+1] - 0.1 d[t] = v[t]$
oh just simplify*

u.
fank
still

$$v[n] = \alpha (s[n] - g[n])$$

$$\alpha [d[n-1] - g[n]]$$

g - s

desired - actual

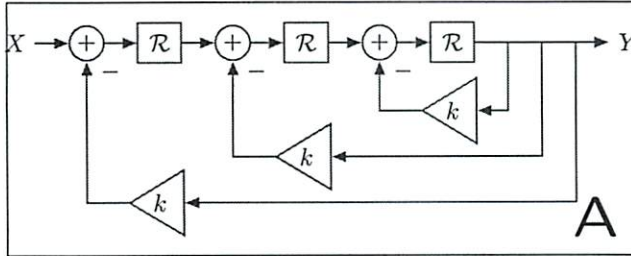
5 Linear Systems (20 points)

think about it

7-5=2
5-7=-2 should be 0

Part a. Several systems are illustrated below. The left column shows block diagrams. The right column shows system functions and pole locations for $k = -0.9$. Indicate which panel on the right (if any) corresponds to each panel on the left by drawing a straight line from A, B, C, and/or D to its partner. [Your answer should include 4 or fewer straight lines.]

can be ±



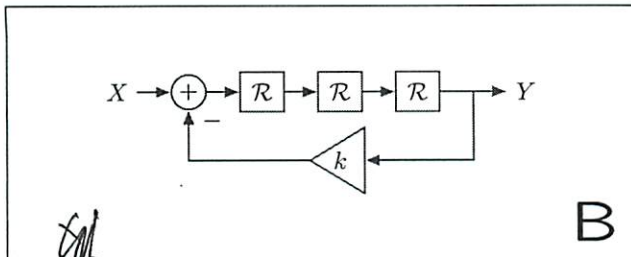
$$\frac{Y}{X} = \frac{\mathcal{R}^3}{k\mathcal{R}^3 + k\mathcal{R}^2 + k\mathcal{R} + 1}$$

poles: 1.72; $-0.41 \pm 0.59j$

E

+ flow in
- flow out

I guess

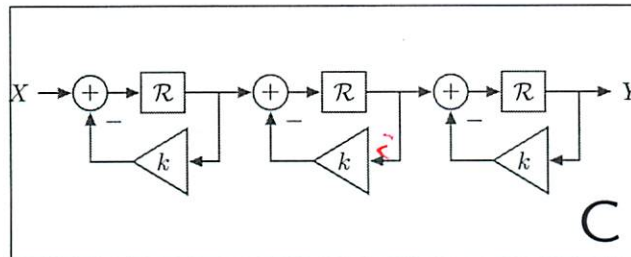


$$\frac{Y}{X} = \frac{1}{k\mathcal{R}^3 + 1}$$

poles: $-0.48 \pm 0.84j$; 0.96

F

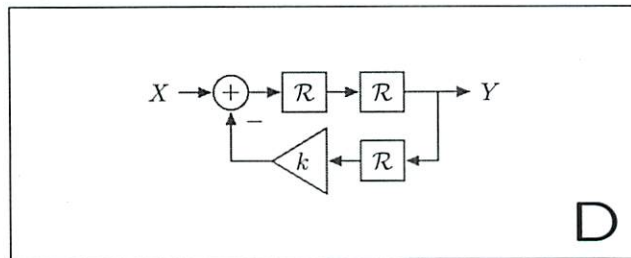
what you define + is
just think about it



$$\frac{Y}{X} = \frac{\mathcal{R}^3}{k\mathcal{R}^3 + 1}$$

poles: $-0.48 \pm 0.84j$; 0.96

G



$$\frac{Y}{X} = \frac{\mathcal{R}^3}{k^3\mathcal{R}^3 + 3k^2\mathcal{R}^2 + 3k\mathcal{R} + 1}$$

poles: 0.9; 0.9; 0.9

H

*Notice that there is a minus sign associated with the bottom input of each adder. This denotes that the input is subtracted (as if k were negative).

6 Python and OOP (20 points)

Part a. What does this program print? Write your answers in the boxes provided.

```
class NN:
    def __init__(self):
        self.n = 0
    def get(self):
        self.n += 1
        return str(self.n)
    def reset(self):
        self.n = 0
class NS(NN):
    def get(self, s):
        return s + NN.get(self)
foo = NS()
print foo.get('a')
```

a1



```
print foo.get('b')
```

b2



```
foo.reset()
print foo.get('c')
```

c1



5 Linear Systems

don't forget

$$B) \quad Y[n] = R^3 \cdot \cancel{x[n]} - k \cdot Y[n-1]$$

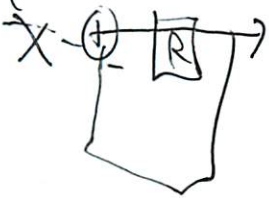
↑
Does the R on top count?
yes

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

$$\frac{Y}{X} = \frac{R^3}{1+R}$$

$$\frac{R^3}{k R^3 + 1}$$

sample

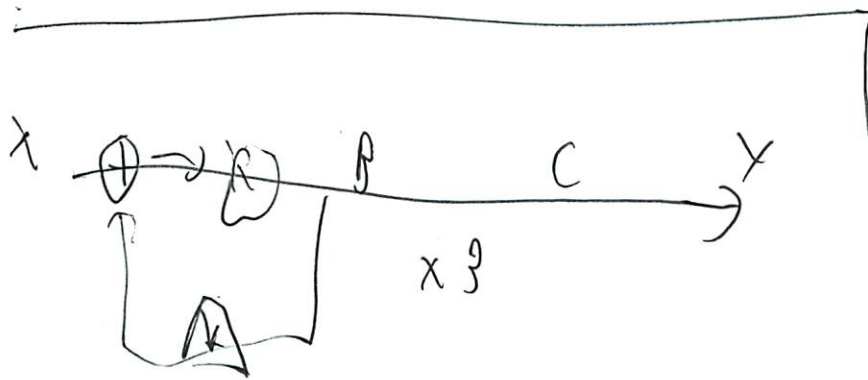


$$y[n] = x[n-1] + y[n-1]$$

↑
y has $\frac{R^0}{R}$ natural

- bottom does not have natural R

- needs to be a R somewhere



think patterns

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

$$B[n] = x[n-1] + k B[n-1]$$

$$C[n] = B[n-1] + k[C[n-1]]$$

$$C[n] = x[n-2] + k \cancel{B}[n-2] + k C[n-1]$$

$$Y[n] = C[n-1] + k Y[n-1]$$

$$Y[n] = x[n-3] + k \cancel{B}[n-3] + \cancel{k} \cancel{C}[n-2] + k Y[n-1]$$

$$Y[n] = x R^3 + k \quad \begin{array}{l} \text{confused w/ } B+C\text{s} \\ \text{- combine} \end{array}$$

$$B = xR + kBR$$

$$B - kBR = xR$$

$$\frac{B}{x} = \frac{\cancel{x}R}{(1-kR)}$$

$$C = BR + kCR$$

$$C - kCR = BR$$

$$\frac{C}{B} = \frac{\cancel{B}R}{(1-k\cancel{R})}$$

$$Y = CR + kYR$$

$$Y - kYR = CR$$

$$\frac{Y}{C} = \frac{\cancel{C}R}{1-kR}$$

Cascade - multiply

- ~~there~~ there is always a form that works

$$\frac{\cancel{B}}{x} \cdot \frac{C}{\cancel{B}} \cdot \frac{Y}{\cancel{C}} = \frac{Y}{x} = \frac{R}{(1-kR)} \cdot \frac{R}{(1-kR)} \cdot \frac{R}{(1-kR)} = \frac{R^3}{(1-kR)^3}$$

Complex

$$1 - 2kR + (kR)^2 = (1-kR)^2$$

$$|s| \cdot (-2kR + k^2R^2 - kR + 2k^2R^2 - k^3R^3)$$

$$+ k^3R^3 + 3k^2R^2 + 3kR + 1$$

close only sign errors

- why does each one only want us to find polts?

How do you do that?

$$Y[n] = xR^2 + ykR^3$$

$$Y + ykR^3 = xR^2$$

$$\frac{Y}{x} = \frac{R^2}{1 + kR^3}$$

~~A~~

~~B~~

no where

↳ so close doesn't count

This is complex as to how

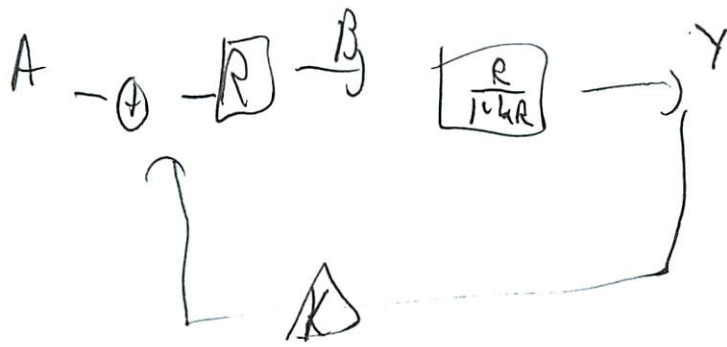
do letters

$$x \quad \begin{array}{cc} A & B \end{array} \quad Y$$

$$Y = BR + kR Y$$

$$\frac{Y}{B} = \frac{R}{1 - kR}$$

Or know feed back subtract thing



$$Y = BR A \left(\frac{R}{1-kR} \right) + kR \left(\frac{R}{1-kR} \right) Y$$

~ yeah I don't know how to do these

You can see it E visually - but how to do

And remember Black's formula

-review most recent design lab

Part b. Write a Python expression that uses `optOverLine` to find the *maximum* of $x^3 - 6$ for x in the range -10 to 10 . Test 200 x values in that range.

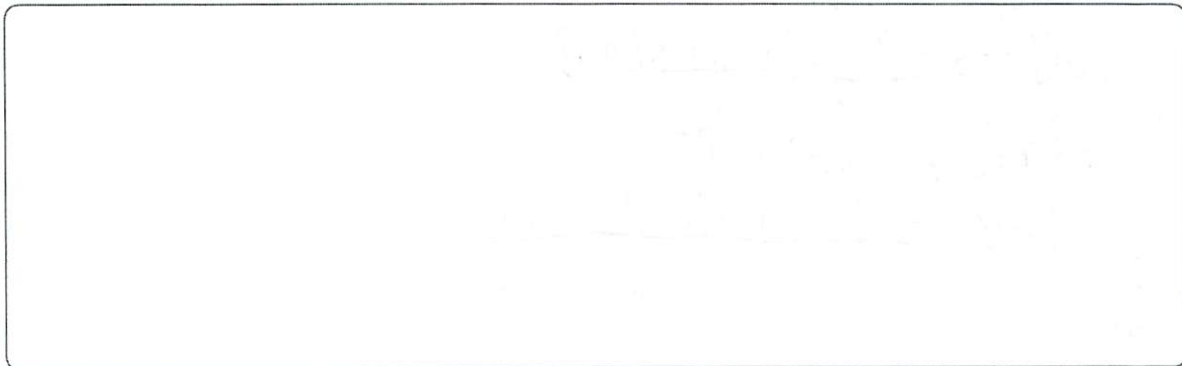
Here is the documentation for `optOverLine`:

```
optOverLine(objective, xmin, xmax, numXsteps, compare=<built-in function lt>)
```

Parameters:

- `objective`: a function that takes a single number as an argument and returns a value
- `compare`: a function from two values (of the type returned by `objective`) to a Boolean; should return True if the first argument is preferred and False otherwise.

Returns: a pair, `(objective(x), x)`, where x is one of the numeric values achieved by starting at `xmin` and taking `numXsteps` equal-sized steps up to `xmax`; the particular value of x returned is the one for which `objective(x)` is best, according to the `compare` operator.



- take home test part

7 System Function Class (20 points)

Consider two system functionals:

$$H_1 = \frac{Y}{X} = \frac{\mathcal{R}^2 + \mathcal{R}}{\mathcal{R} + 3}$$

and

$$H_2 = \frac{Y}{X} = \frac{\mathcal{R}^3 + \mathcal{R}^2}{\mathcal{R}^2 + 3\mathcal{R}}$$

Part a. Determine the corresponding difference equations.

Difference equation corresponding to H_1 :

$$Y(\mathcal{R} + 3) = \mathcal{R}^2 X + \mathcal{R} X$$

$$Y[n] = -Y[n-1] + X[n-2] + X[n-1]$$

Difference equation corresponding to H_2 :

$$\mathcal{R}^2 Y + 3\mathcal{R} Y = X\mathcal{R}^3 + X\mathcal{R}^2$$

$$Y[n-1] = -Y[n-2] + X[n-3] + X[n-2]$$

Briefly describe the implications of these difference equations for similarities and differences between the two systems.

Oh same

Part b. The numerators and denominators of H_1 and H_2 contain factors of \mathcal{R} :

$$H_1 = \frac{Y}{X} = \frac{\mathcal{R}^2 + \mathcal{R}}{\mathcal{R} + 3} = \frac{\mathcal{R} \times (\mathcal{R} + 1)}{\mathcal{R} + 3}$$

$$H_2 = \frac{Y}{X} = \frac{\mathcal{R}^3 + \mathcal{R}^2}{\mathcal{R}^2 + 3\mathcal{R}} = \frac{\mathcal{R} \times \mathcal{R} \times (\mathcal{R} + 1)}{\mathcal{R} \times (\mathcal{R} + 3)}$$

If we cancel one \mathcal{R} factor from the numerator and denominator of H_2 , then $H_2 = H_1$.

what I said
- did not know you could do it like that

Write a Python function called `ReducedSystemFunction`, which

- takes an input `inp` of type `SystemFunction`,
- cancels *all* factors of \mathcal{R} that are common to the numerator and denominator of `inp`, and
- returns the result as a new object of type `SystemFunction`.

`ReducedSystemFunction` should not modify `inp`. Some relevant software documentation follows the box. You can assume that we have already `import sf` and `poly`.

```
def ReducedSystemFunction(inp):
```

ham - tough one

- look at last denom position - if 0
- move everything to the right (top + bottom)
- repeat (recursive!) till last denom position != 0

Attributes of SystemFunction Class:

<code>__init__(self, numPoly, denomPoly)</code>	
<code>poles(self)</code>	returns a list of the poles of the system
<code>poleMagnitudes(self)</code>	returns a list of the magnitudes of the poles of the system
<code>dominantPole(self)</code>	returns the pole with the largest magnitude
<code>__add__(self, other)</code>	
<code>numerator</code>	Polynomial in \mathcal{R} representing the numerator
<code>denominator</code>	Polynomial in \mathcal{R} representing the denominator

Attributes of Polynomial Class:

<code>__init__(self, coeffs)</code>	
<code>add(p1, p2)</code>	returns a new polynomial, which is sum
<code>__add__(p1, p2)</code>	
<code>__sub__(p1, p2)</code>	
<code>scalarMult(self, s)</code>	returns a new polynomial, with coefs multiplied by s
<code>mul(p1, p2)</code>	returns a new polynomial equal to the product
<code>__mul__(p1, p2)</code>	
<code>val(self, x)</code>	returns value of polynomial with variable assigned to x
<code>roots(self)</code>	return list of roots
<code>coeffs</code>	list of polynomial coefficients, highest order first
<code>order</code>	polynomial order; one less than number of coeffs

8 Inheritance and State Machines (20 points)

Recall that we have defined a Python class `sm.SM` to represent state machines. Here we consider a special type of state machine, whose states are always integers that start at 0 and increment by 1 on each transition. We can represent this new type of state machines as a Python subclass of `sm.SM` called `CountingStateMachine`.

We wish to use the `CountingStateMachine` class to define new subclasses that each provide a single new method `getOutput(self, state, inp)` which returns *just the output* for that state and input; the `CountingStateMachine` will take care of managing and incrementing the state, so its subclasses don't have to worry about it.

Here is an example of a subclass of `CountingStateMachine`.

```
class CountMod5(CountingStateMachine):
    def getOutput(self, state, inp):
        return state % 5
```

get Next Value

Instances of `CountMod5` generate output sequences of the form 0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0, ...

Part a. Define the `CountingStateMachine` class. Since `CountingStateMachine` is a subclass of `sm.SM`, you will have to provide definitions of the `startState` instance variable and `getNextValues` method, just as we have done for other state machines. You can assume that every subclass of `CountingStateMachine` will provide an appropriate `getOutput` method.

```
class CountingStateMachine(sm.SM):
```

-- init =

** getNextValues(self, state, inp)*

```
    return state + 1, state + 1
           ↓       ↑
           state  output
```


Part b. Define a subclass of `CountingStateMachine` called `AlternateZeros`. Instances of `AlternateZeros` should be state machines for which, on even steps, the output is the same as the input, and on odd steps, the output is 0. That is, given inputs, $i_0, i_1, i_2, i_3, \dots$, they generate outputs, $i_0, 0, i_2, 0, \dots$

```
class AlternateZeros(CountingStateMachine):
```

```
    get next value (self, state, inp)
        if state % 2 == 0
            return state + 1, inp
        else
            return state + 1, 0
```

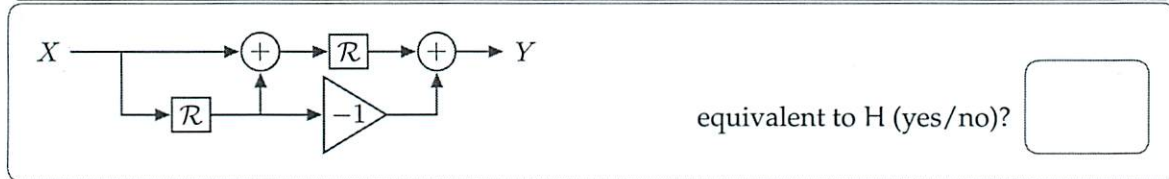
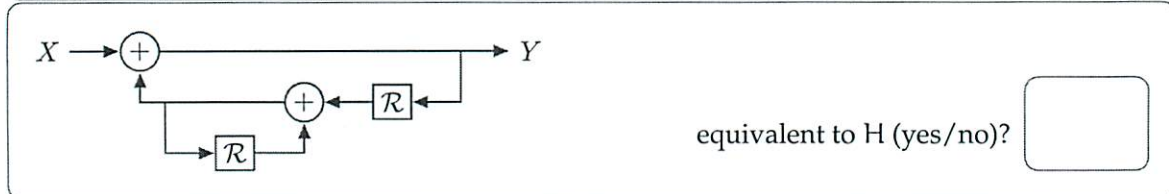
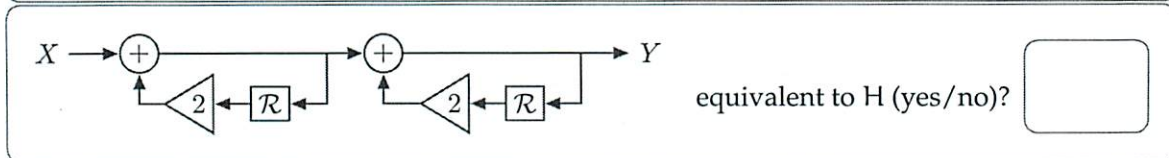
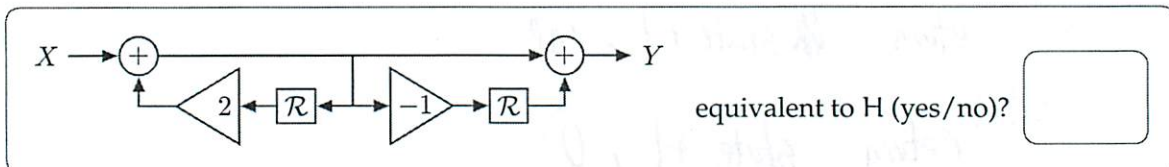
9 Signals and Systems (20 points)

Let H represent a system whose input is a signal X and whose output is a signal Y . The system H is defined by the following difference equations:

$$y[n] = x[n] + z[n]$$

$$z[n] = y[n - 1] + z[n - 1]$$

Part a. Which of the following systems are valid representations of H ? (Remember that there can be multiple “equivalent” representations for a system.)



Part b. Assume that the system starts “at rest” and that the input signal X is the unit sample signal. Determine $y[3]$.

$y[3] =$

Part c. Let p_0 represent the dominant pole of H . Determine p_0 .

Enter p_0 or **none** if there are no poles:

Fall 09

10/11
OH

8. Oh using getOutput

↑ for a given state

So just increment state

- I see then in pt B they implement
a special getOutput

- It's just the wording you need

- Counting SM is abstract

- must subclass it

- and define getOutputs

Fall 04

Midterm 1 Practice Solutions

Says spring 09

in this one's header

1 Robot behavior

1.

```
class TTurnSmoothly(SM):
    def getNextValues(self, state, inp):
        currentTheta = inp.odometry.theta
        return (state,
                io.Action(rvel = self.rotationalGain * (thetaDesired -
                                                            currentTheta),
                           fvel = 0))
```

2. The difference equations are:

$$\Theta[n] = \Theta[n-1] + T\Omega[n-1]$$

$$\Omega[n] = K(\Theta_{des}[n-1] - \Theta[n-1])$$

The operator version is:

$$(1 - \mathcal{R})\Theta = T\mathcal{R}\Omega$$

$$\Omega = K\mathcal{R}(\Theta_{des} - \Theta)$$

Combining we get:

$$1 - \mathcal{R}\Theta + KTR^2\Theta = KTR^2\Theta_{des}$$

We accepted a number of variations.

3. The system function is:

$$\frac{\Theta}{\Theta_{des}} = \frac{KTR^2}{1 - \mathcal{R}\Theta + KTR^2}$$

Substitute $\mathcal{R} = 1/z$ in the denominator polynomial and we get:

$$z^2 - z + KT$$

The roots of this polynomial are the poles:

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

Note that for $K = 0$, one of the poles is 1.0. For $K < 0$, one of the poles is always greater than 1.0. This makes sense since for $K < 0$, the feedback increases the error.

For $T = 0.2$, for $K = 5$, we have a pole of magnitude 1.0 and they get bigger with bigger gain. The range of stable K is $0 < K < 5$

In general, when $4KT > 1$ we have complex poles. For monotonic convergence we need real poles, so we want:

$$0 < K \leq 1/(4T)$$

Note that when $K = 1/(4T)$, we have the lowest magnitude pole, the pole approaches 1 as K decreases towards 0, so:

For $T = 0.2$, the best value of K is 1.25, which leads to a pole of 1/2.

4. The analysis above shows that when T increases the range of stable gains decreases. Basically the effective gain is KT . As the T increases we need smaller and smaller gains to keep monotonic convergence. The result is very sluggish performance.
- 5.

$$\Theta[n] = \Theta[n-1] + T\Omega[n-1]$$

$$\Omega[n] = \Omega[n-1] + T\Xi[n-1]$$

$$\Xi[n] = K_1 E[n] + K_2 E[n-1]$$

$$E[n] = \Theta_{des}[n] - \Theta[n]$$

where Ξ is the acceleration. So, basically, the velocity is the integrated acceleration and the position is the integrated velocity. And, the acceleration is given by the gains times the position errors.

```
def robot(k1, k2):
    dt = 0.2
    control = sf.SystemFunction(poly.Polynomial([k2, k1]),
                                poly.Polynomial([1]))
    vel = sf.SystemFunction(poly.Polynomial([T, 0]),
                            poly.Polynomial([-1, 1]))
    pos = sf.SystemFunction(poly.Polynomial([T, 0]),
                            poly.Polynomial([-1, 1]))
    rob = sf.FeedbackSubtract(sf.Cascade(control,
                                          sf.Cascade(vel, pos)))
    return abs(rob.dominantPole())
```

Depending on the resolution of the search, one gets very different answers.

```
optOverGrid(robot, -10, 10, -10, 10, 1, 1)
(0.9999999999999999, (1, -1))
optOverGrid(robot, -10, 10, -10, 10, 0.5, 0.5)
(0.84617988641100905, (2.5, -2.0))
optOverGrid(robot, -10, 10, -10, 10, 0.25, 0.25)
(0.7499999999999999, (1.75, -1.5))
optOverGrid(robot, -10, 10, -10, 10, 0.1, 0.1)
(0.68863445766586584, (1.4999999999999998, -1.3000000000000001))
```

2 Weighted Objective

1.

There are many ways of writing this. Here are a few.

```
def argmax(elements, f):
    bestScore = None
    bestElement = None
    for e in elements:
        score = f(e)
        if bestScore == None or score > bestScore:
            bestScore = score
            bestElement = e
    return bestElement
```

```
def argmax(elements, f):
    bestElement = elements[0]
    for e in elements:
        if f(e) > f(bestElement):
            bestElement = e
    return bestElement
```

```
def argmax(elements, f):
    vals = [f(e) for e in elements]
    return elements[vals.index(max(vals))]
```

```
def argmax(elements, f):
    return max(elements, key=f)
```

2.

Here are a couple that work:

```
WOPQ([Fish.length, Fish.width], [0.9, 0.1])
WOPQ([lambda x: x.length(), lambda x: x.width()], [0.9, 0.1])
```

3.

```
def extract(self):
    def priority(e):
        return sum([w*f(e) for (w,f) in zip(self.weights,self.features)])
    best = argmax(self.elements, priority)
    self.elements.remove(best)
    return best
```

4.

```
wopq.extract() = Sydney
```

Sydney is the longest fish and the priority weights heavily favor length.

5.

```
class WOPQ(FPQ):
    def __init__(self, features, weights):
        def priority(e):
            return sum([w*f(e) for (w,f) in zip(weights,features)])
        FPQ.__init__(self, priority)
```

That's it... the other methods are provided by the FPQ class.

6.

```
def featureRange(feature, elements):
    vals = [feature(e) for e in elements]
    return max(vals) - min(vals)

def selectFeatures(features, n, elements):
    return sorted(features, key = featureRange)[-n : ]
```

3 More Oop

You are told that we have a class A, with one initialization argument, with a method f that takes one argument and returns a number.

```
>>> x = A(2)
>>> x.f(3)
9
```

Define a class B that is a subclass of class A. Class B has the same initialization as A and a method f that returns twice whatever value the f method of A would return. You don't need to know what the code for A is.

```
>>> x = B(2)
>>> x.f(3)
18
```

```
class B(A):
    def f(self, x):
        return A.f(self, x)*2
```

2 points for correct answer (no `__init__`, passing self)

1 point for minor error

4 In the Tank

Imagine a cylindrical water tank with a source of water entering the bottom of the tank and a sensor to measure the depth of the water. The source of water can be adjusted so that water enters (or exits) the tank at a desired rate. In addition to this controlled source of water, there is also a leak in the tank, through which water flows regardless of the controlled flow. The diameter of the tank is such that one inch in depth is one gallon of water. The following variables characterize this problem:

- $d[t]$ is the depth, in inches, of the water in the tank at time t .
- $v[t]$ is the volume of water that flows through the controlled source of water in the time interval $[t, t + 1]$.
- $o[t]$ is the volume of water that leaks from the tank in the time interval $[t, t + 1]$. Assume $o[t]$ is proportional to $d[t]$, with a proportionality constant of 0.1
- $g[t]$ is the desired (goal) depth of water (in inches) in the tank at time t
- $s[t]$ is the sensed depth (in inches) of water in the tank at time t ; assume the sensor introduces one unit of delay, so that $s[t] = d[t - 1]$

a. Write a difference equation that describes the depth of water in the tank ($d[\cdot]$) in terms of the flow through the controlled source ($v[\cdot]$).

$$d[t + 1] = d[t] + v[t] - o[t]$$

$$o[t] = 0.1d[t]$$

$$d[t + 1] - 0.9d[t] = v[t]$$

3 points if correct

2 points if expression contains $o[t]$ but otherwise correct

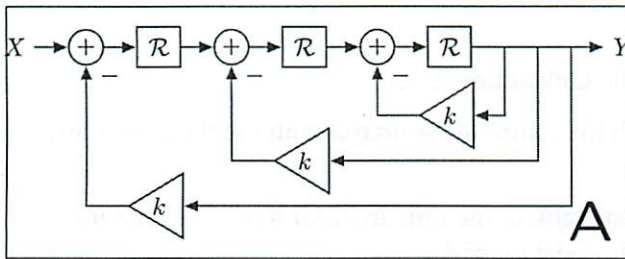
b. Write a difference equation describing a proportional controller that specifies $v[\cdot]$ as a function of sensed depth $s[\cdot]$ and desired depth $g[\cdot]$. Let α represent the constant of proportionality.

$$v[t] = \alpha(g[t] - s[t])$$

3 points if correct

5 Linear Systems (20 points)

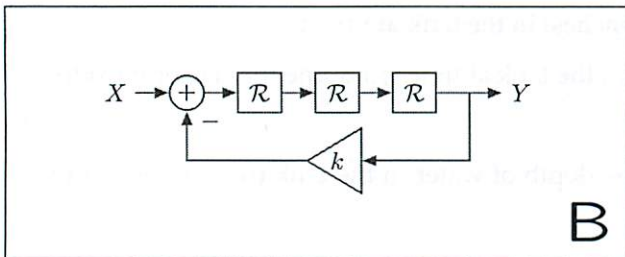
Part a. Several systems are illustrated below. The left column shows block diagrams. The right column shows system functions and pole locations for $k = 0.9$. Indicate which panel on the right (if any) corresponds to each panel on the left by drawing a straight line from A, B, C, and/or D to its partner. [Your answer should include 4 or fewer straight lines.]



$$\frac{Y}{X} = \frac{\mathcal{R}^3}{k\mathcal{R}^3 + k\mathcal{R}^2 + k\mathcal{R} + 1}$$

poles: 1.72; $-0.41 \pm 0.59j$

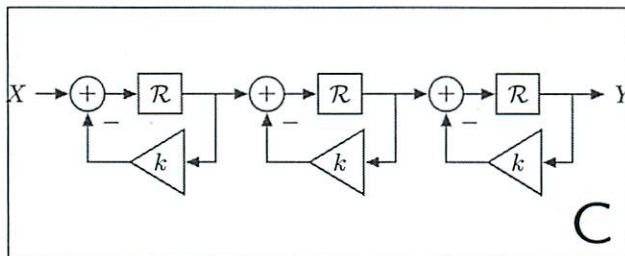
E



$$\frac{Y}{X} = \frac{1}{k\mathcal{R}^3 + 1}$$

poles: $-0.48 \pm 0.84j$; 0.96

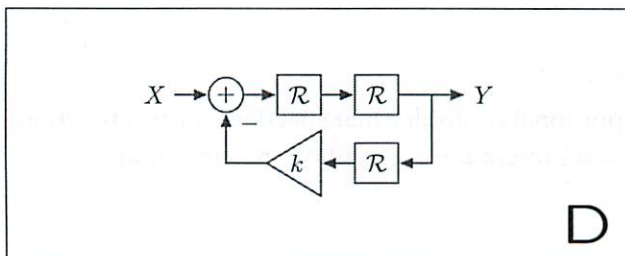
F



$$\frac{Y}{X} = \frac{\mathcal{R}^3}{k\mathcal{R}^3 + 1}$$

poles: $-0.48 \pm 0.84j$; 0.96

G



$$\frac{Y}{X} = \frac{\mathcal{R}^3}{k^3\mathcal{R}^3 + 3k^2\mathcal{R}^2 + 3k\mathcal{R} + 1}$$

poles: 0.9; 0.9; 0.9

H

- A → E → +3 points if correct
- B → G → +3 points if correct
- C → H → +3 points if correct
- D → nowhere → +2 points if correct

6 Python and OOP (20 points)

Part a. What does this program print? Write your answers in the boxes provided.

```
class NN:
    def __init__(self):
        self.n = 0
    def get(self):
        self.n += 1
        return str(self.n)
    def reset(self):
        self.n = 0
class NS(NN):
    def get(self, s):
        return s + NN.get(self)
foo = NS()
print foo.get('a')
```

a1

```
print foo.get('b')
```

b2

```
foo.reset()
print foo.get('c')
```

c1

Part b. Write a Python expression that uses `optOverLine` to find the *maximum* of $x^3 - 6$ for x in the range -10 to 10 . Test 200 x values in that range.

Here is the documentation for `optOverLine`:

```
optOverLine(objective, xmin, xmax, numXsteps, compare=<built-in function lt>)
```

Parameters:

- `objective`: a function that takes a single number as an argument and returns a value
- `compare`: a function from two values (of the type returned by `objective`) to a Boolean; should return `True` if the first argument is preferred and `False` otherwise.

Returns: a pair, `(objective(x), x)`, where x is one of the numeric values achieved by starting at `xmin` and taking `numXsteps` equal-sized steps up to `xmax`; the particular value of x returned is the one for which `objective(x)` is best, according to the `compare` operator.

```
return optOverLine(lambda x: x**3 - 6, -10, 10, 200, operator.gt)[0]
```

7 SystemFunction Class (20 points)

Consider two system functionals:

$$H_1 = \frac{Y}{X} = \frac{\mathcal{R}^2 + \mathcal{R}}{\mathcal{R} + 3}$$

and

$$H_2 = \frac{Y}{X} = \frac{\mathcal{R}^3 + \mathcal{R}^2}{\mathcal{R}^2 + 3\mathcal{R}}$$

Part a. Determine the corresponding difference equations.

Difference equation corresponding to H_1 :

$$y[n] = -\frac{1}{3}y[n-1] + \frac{1}{3}x[n-1] + \frac{1}{3}x[n-2]$$

Difference equation corresponding to H_2 :

$$y[n-1] = -\frac{1}{3}y[n-2] + \frac{1}{3}x[n-2] + \frac{1}{3}x[n-3]$$

Briefly describe the implications of these difference equations for similarities and differences between the two systems.

H_2 is equivalent to H_1 with an extra pole and an extra zero, both at $z = 0$. Because these poles cancel, the systems will give identical responses to identical inputs; to see this, substitute $n - 1$ for n in the first equation. (Note: as an example of such a pair of systems, imagine two identical state machines being fed the same input, but one gets the input one time step later than the other. Clearly both generate the same output for a given input, but in the second one both the input and output are delayed by one time step.)

Part b. The numerators and denominators of H_1 and H_2 contain factors of \mathcal{R} :

$$H_1 = \frac{Y}{X} = \frac{\mathcal{R}^2 + \mathcal{R}}{\mathcal{R} + 3} = \frac{\mathcal{R} \times (\mathcal{R} + 1)}{\mathcal{R} + 3}$$

$$H_2 = \frac{Y}{X} = \frac{\mathcal{R}^3 + \mathcal{R}^2}{\mathcal{R}^2 + 3\mathcal{R}} = \frac{\mathcal{R} \times \mathcal{R} \times (\mathcal{R} + 1)}{\mathcal{R} \times (\mathcal{R} + 3)}$$

If we cancel one \mathcal{R} factor from the numerator and denominator of H_2 , then $H_2 = H_1$.

Write a Python function called `ReducedSystemFunction`, which

- takes an input `inp` of type `SystemFunction`,
- cancels *all* factors of \mathcal{R} that are common to the numerator and denominator of `inp`, and
- returns the result as a new object of type `SystemFunction`.

`ReducedSystemFunction` should not modify `inp`. Some relevant software documentation follows the box. You can assume that we have already import `sf` and `poly`.

```
def ReducedSystemFunction(inp):
    num= inp.numerator.coeffs[:]
    den= inp.denominator.coeffs[:]
    while num[-1] == 0 and den[-1] == 0:
        num.pop()
        den.pop()
    return sf.SystemFunction(poly.Polynomial(num),
                             poly.Polynomial(den))
```

Attributes of SystemFunction Class:

<code>__init__(self, numPoly, denomPoly)</code>	
<code>poles(self)</code>	returns a list of the poles of the system
<code>poleMagnitudes(self)</code>	returns a list of the magnitudes of the poles of the system
<code>dominantPole(self)</code>	returns the pole with the largest magnitude
<code>__add__(self, other)</code>	
<code>numerator</code>	Polynomial in \mathcal{R} representing the numerator
<code>denominator</code>	Polynomial in \mathcal{R} representing the denominator

Attributes of Polynomial Class:

<code>__init__(self, coeffs)</code>	
<code>add(p1, p2)</code>	returns a new polynomial, which is sum
<code>__add__(p1, p2)</code>	
<code>__sub__(p1, p2)</code>	
<code>scalarMult(self, s)</code>	returns a new polynomial, with coefs multiplied by s
<code>mul(p1, p2)</code>	returns a new polynomial equal to the product
<code>__mul__(p1, p2)</code>	
<code>val(self, x)</code>	returns value of polynomial with variable assigned to x
<code>roots(self)</code>	return list of roots
<code>coeffs</code>	list of polynomial coefficients, highest order first
<code>order</code>	polynomial order; one less than number of coeffs

8 Inheritance and State Machines (20 points)

Recall that we have defined a Python class `sm.SM` to represent state machines. Here we consider a special type of state machine, whose states are always integers that start at 0 and increment by 1 on each transition. We can represent this new type of state machines as a Python subclass of `sm.SM` called `CountingStateMachine`.

We wish to use the `CountingStateMachine` class to define new subclasses that each provide a single new method `getOutput(self, state, inp)` which returns *just the output* for that state and input; the `CountingStateMachine` will take care of managing and incrementing the state, so its subclasses don't have to worry about it.

Here is an example of a subclass of `CountingStateMachine`.

```
class CountMod5(CountingStateMachine):
    def getOutput(self, state, inp):
        return state % 5
```

Instances of `CountMod5` generate output sequences of the form 0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0, ...

Part a. Define the `CountingStateMachine` class. Since `CountingStateMachine` is a subclass of `sm.SM`, you will have to provide definitions of the `startState` instance variable and `getNextValues` method, just as we have done for other state machines. You can assume that every subclass of `CountingStateMachine` will provide an appropriate `getOutput` method.

```
class CountingStateMachine(sm.SM):
    def __init__(self):
        self.startState = 0
    def getNextState(self, state, inp):
        return (state + 1, self.getOutput(state, inp))
```

Part b. Define a subclass of `CountingStateMachine` called `AlternateZeros`. Instances of `AlternateZeros` should be state machines for which, on even steps, the output is the same as the input, and on odd steps, the output is 0. That is, given inputs, $i_0, i_1, i_2, i_3, \dots$, they generate outputs, $i_0, 0, i_2, 0, \dots$

```
class AlternateZeros(CountingStateMachine):
    def getOutput(self, state, inp):
        if not state % 2:
            return inp
        return 0
```

9 Signals and Systems (20 points)

Let H represent a system whose input is a signal X and whose output is a signal Y . The system H is defined by the following difference equations:

$$y[n] = x[n] + z[n]$$

$$z[n] = y[n - 1] + z[n - 1]$$

Part a. Which of the following systems are valid representations of H ? (Remember that there can be multiple “equivalent” representations for a system.)

	equivalent to H (yes/no)? YES
	equivalent to H (yes/no)? NO
	equivalent to H (yes/no)? YES
	equivalent to H (yes/no)? NO

Part b. Assume that the system starts “at rest” and that the input signal X is the unit sample signal. Determine $y[3]$.

$y[3] =$ 4

Part c. Let p_0 represent the dominant pole of H . Determine p_0 .

Enter p_0 or **none** if there are no poles: 2

6.01 Midterm 1: Fall 2009

Name: Solutions

Section:

Enter all answers in the boxes provided.

During the exam you may:

- read any paper that you want to
- use a laptop, but **only to READ course material**

You may not

- search the web generally
- run Idle or Python
- talk, chat or email or otherwise converse with another person
- use a music player

Because of this midterm, we will not have software labs this week. Instead, we will provide lectures on control systems during your software lab time slot:

section 1	October 6	11:00am	26-100
section 2	October 6	2pm	34-101

For staff use:

1.	/10
2.	/20
3.	/20
4.	/10
5.	/20
6.	/20
total:	/100

6.01 Midterm 1: Fall 2009

Name:	Section:
--------------	-----------------

Enter all answers in the boxes provided.

During the exam you may:

- read any paper that you want to
- use a laptop, but **only to READ course material**

You may not

- search the web generally
- run Idle or Python
- talk, chat or email or otherwise converse with another person
- use a music player

Because of this midterm, we will not have software labs this week. Instead, we will provide lectures on control systems during your software lab time slot:

section 1	October 6	11:00am	26-100
section 2	October 6	2pm	34-101

For staff use:

1.	/10
2.	/20
3.	/20
4.	/10
5.	/20
6.	/20
total:	/100

1 OOP (10 points)

The following definitions have been entered into the Python shell:

```
class F:
    x = 0
    y = 10
    def __init__(self, y):
        self.x = y
    def a1(self, y):
        self.x = max(self.x, y)
        return self.x
    def a2(self):
        self.x = max(self.x, self.y)
        return self.x
class G(F):
    def b(self):
        self.y = self.x * self.x
        return self.y
```

Output

Write the values of the following expressions (enter None when there is no value; write Error when an error results and explain briefly why it's an error). Assume these expressions are evaluated one after the other (in each column).

$x=5$	<code>f = F(5)</code> None ✓	<code>g = G()</code> Error → no init parameter ✓	<i>new instance</i>
$x=7$	<code>f.a1(7)</code> 7 ✓	<code>g = G(3)</code> None ✓	$x=3$
$x=7$	<code>f.a1(3)</code> 7 ✓	<code>g.a1(5)</code> 5 ✓	$x=5$
	<code>f.b()</code> Error no such method ✓ <i>↑ was thinking - write it</i>	<code>g.b()</code> 25 ✓	$y=25$
$x=10$	<code>f.a2()</code> 10 ✓	<code>g.a2()</code> 25 ✓	

super!

2 Signals (20 points)

Given a signal x , we want to construct a signal y such that:

$$y[n] = a \cdot x[n-1] + (1-a) \cdot x[n-2]$$

$$ax[n-1] + (1-a)x[n-2]$$

for $0 \leq a \leq 1$.

Part a. Define a `Signal` subclass that represents this signal, given an instance of the `Signal` class representing x and the value of a . Construct the new signal, given a signal instance x , and a variable a , as follows:

$y = \text{MySig}(x, a)$

Do not use any of the existing `Signal` subclasses, such as `Rn` or `PolyR`.

```
class MySig(sig.Signal):
    sum (gain a, on x, value(a), unit signal)
    subtract (x.value(a-2), gain a(x.value(a-2)))
    or something like that
    - would refine on test
```

$\text{self.a} \cdot \text{self.x.sample}(n-1) +$
 $(1-\text{self.a}) \cdot \text{self.x.sample}(n-2)$
 (don't forget)

Part b. Given a signal instance x , and a variable a , write an expression involving `sig.PolyR` that constructs an equivalent signal. To construct a polynomial use `poly.Polynomial`.

```
sig.PolyR(poly.Polynomial([1-a, a, 0], sig))
```

look up R2 in front

can you do that?

sure

oh were supposed to do this step and not gain

oh I was going to do that

w/ unit signal?

Part c. Write a Python procedure `settleTime` that is given:

- `s`: an instance of the `Signal` class;
- `m`: an integer, representing the max index to look at;
- `bounds`: a tuple of two values (`lo`, `hi`).

It returns an integer or `None`. The integer corresponds to the smallest sample index (`w`) such that the sample at index `w` and all the samples from `w` up to `m` are between `lo` and `hi` (less than or equal to `hi` and greater or equal to `lo`). It returns `None` if no such sample exists.

```
def settleTime(s, m, bounds):
```

4 Difference Equations (10 points)

Newton's law of cooling states that:

The change in an object's temperature from one time step to the next is proportional to the difference (on the earlier step) between the temperature of the object and the temperature of the environment, as well as to the length of the time step.

Let

- $o[n]$ be temperature of object
- $s[n]$ be temperature of environment
- T be the duration of a time step
- K be the constant of proportionality

Part a. Write a difference equation for Newton's law of cooling. Be sure the signs are such that the temperature of the object will eventually equilibrate with that of the environment.

$$o[n] =$$

Part b. Write the system function corresponding to this equation (show your work):

$$H = \frac{O}{S} =$$

3 State Machines (20 points)

Write a state machine whose inputs are the characters of a string of “words” separated by spaces. For each input character, the machine should:

- output the string 'x' if the character is within a word, or
- if the input character is a space, then
 - output the most recent word if the most recent word is a number (all numerical digits), or
 - output None if the most recent input word is empty or contains any non-digits.

Assume there is never more than one consecutive space. For example:

```
>>> x = ' v1 11 1v '  
>>> FindNumbers().transduce(x)  
[None, 'x', 'x', None, 'x', 'x', '11', 'x', 'x', None]
```

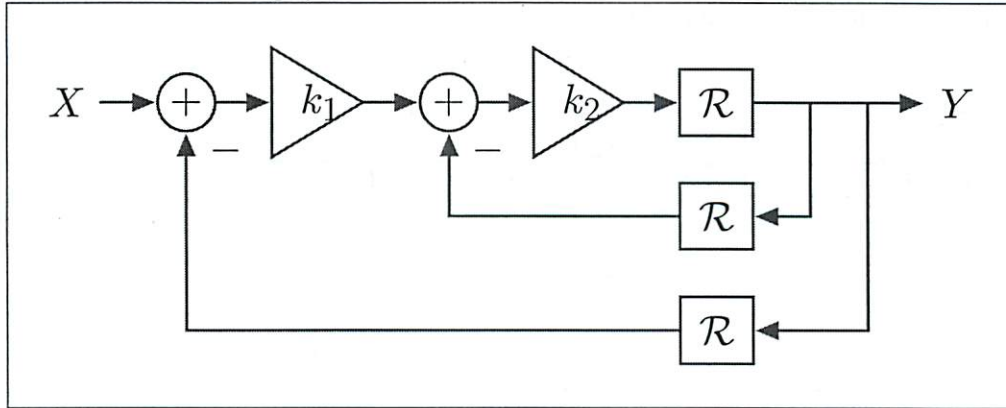
Do not add any instance attributes to the state machines.

If `foo` is a single-character or multi-character string, the Python method `foo.isdigit()` returns True if there is at least one character in `foo` and all of the characters in `foo` are all digits.

```
class FindNumbers(sm.SM):
```

5 Signals and Systems (20 points)

Consider the following system:



Part a. Write the system function:

$$H = \frac{Y}{X} =$$

Part b.

Let $k_1 = 1$ and $k_2 = -2$. Assume that the system starts "at rest" (all signals are zero) and that the input signal X is the unit sample signal. Determine $y[0]$ through $y[3]$.

$y[0] =$

$y[1] =$

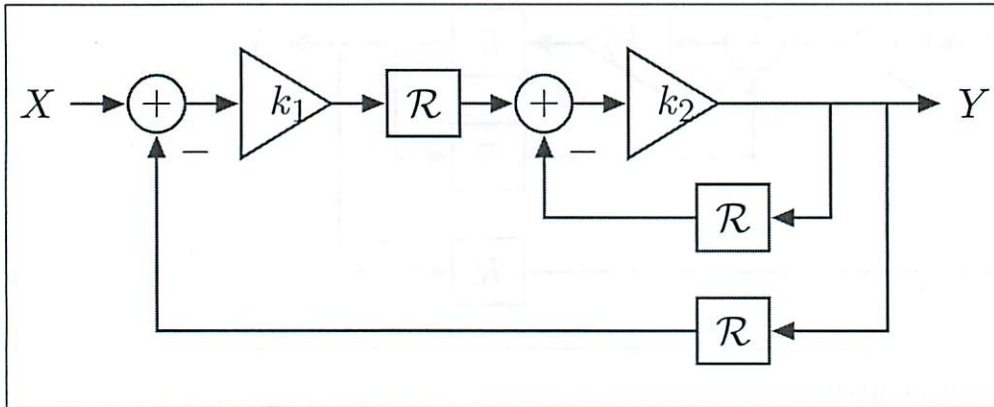
$y[2] =$

$y[3] =$

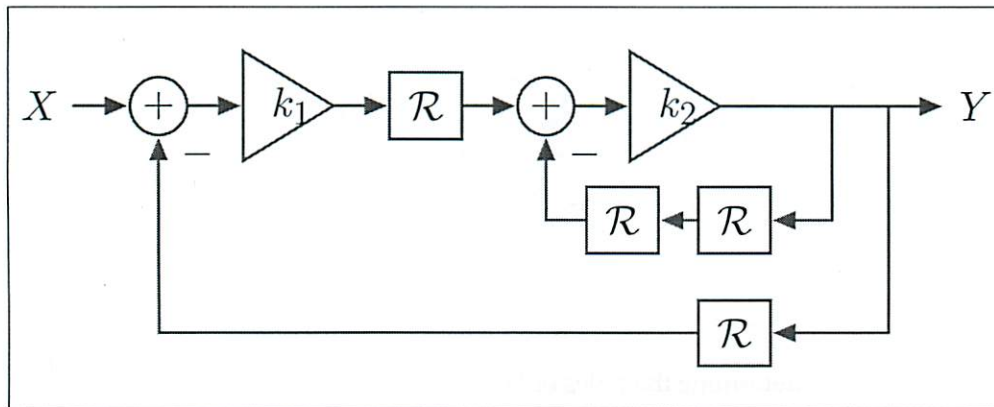
Part c. Let $k_1 = 1$ and $k_2 = -2$, determine the poles of H .

Enter poles or **none** if there are no poles:

Part d. For each of the systems below indicate whether the system is equivalent to this one: (Remember that there can be multiple “equivalent” representations for a system.) If you write clearly the system function for these systems, we may be able to give you partial credit.



Equivalent to H (yes/no)?

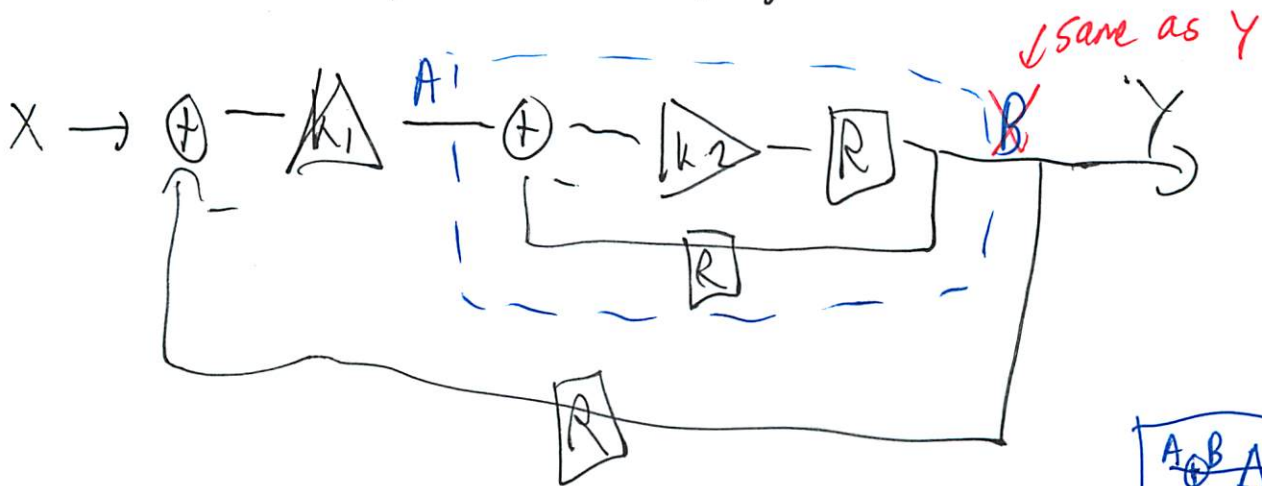


Equivalent to H (yes/no)?

Fall 09 #5 Signals + Systems

10/11
OH

- same idea, more complex to Spring 2018 #7



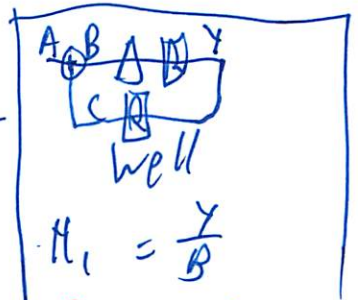
Once again break up

$$H = \frac{Y}{A}$$

$$H_1 = k_2 R A \leftarrow \text{implicitly } \frac{Y}{A}$$

$$H_2 = R X$$

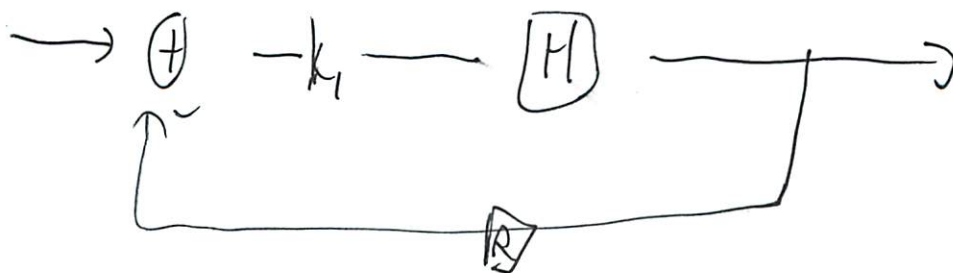
all taken care of $H_2 = \frac{C}{Y}$



$$\frac{H_1}{1 + H_1 H_2} = \frac{k_2 R A}{1 + (k_2 R A)(R X)} = \frac{Y}{A}$$

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

Add another part



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

②

Now whole thing

$$H_1 = \frac{k_1 k_2 R}{1 + k_2 R^2}$$

$$H_2 = R$$

$$\frac{Y}{X} = \frac{k_1 k_2 R}{1 + k_2 R^2} \div \left(1 + \left(\frac{k_1 k_2 R}{1 + k_2 R^2} \right) \left(\frac{R}{1} \right) \right)$$

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

$$1 + k_2 R^2$$

$$1 + k_2 R^2$$

simplify are more

multiply by

-I would have flipped should be same

3

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

$$(1 + k_2 R^2)(k_1 k_2 R^2)$$

They factor a portion
make into a polynomial of R

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

$$R^2(k_1 k_2 + k_2) + 1$$

part b

Run the SM
plug in

$$1 \cdot -2 R$$

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

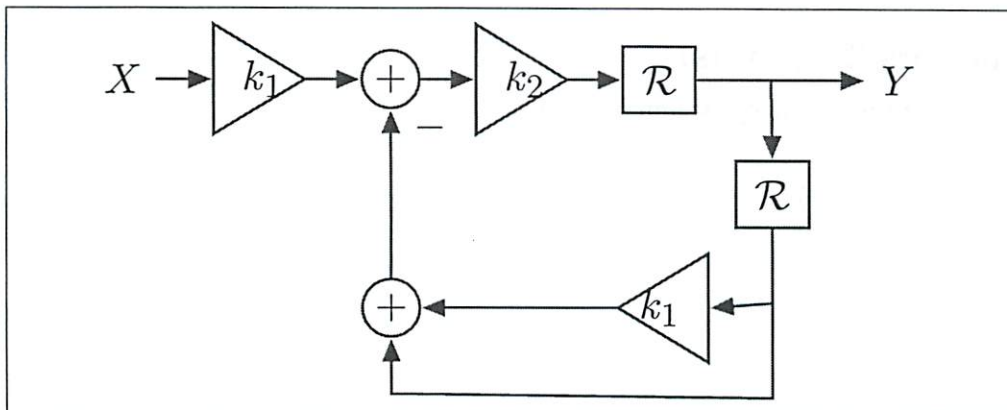
simplify

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

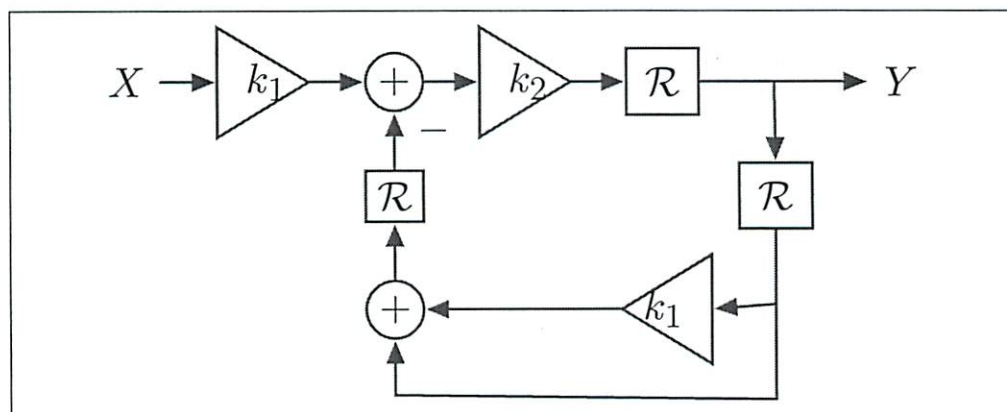
- can translate it to diff eq to make it easier

$$Y(R^2(k_1 k_2 + k_2) + 1) = X k_1 k_2 R$$

$$Y[n] = -Y[n-2](k_1 k_2 + k_2) + X[n-1]k_1 k_2$$



Equivalent to H (yes/no)?



Equivalent to H (yes/no)?

6 System Behaviors (20 points)

Part a. Find the poles for the following system functions:

$$H_1(\mathcal{R}) = \frac{1}{1 - \mathcal{R} + 0.5\mathcal{R}^2}$$

Poles:

$$H_2(\mathcal{R}) = \frac{1}{1 - 0.4\mathcal{R} - 0.05\mathcal{R}^2}$$

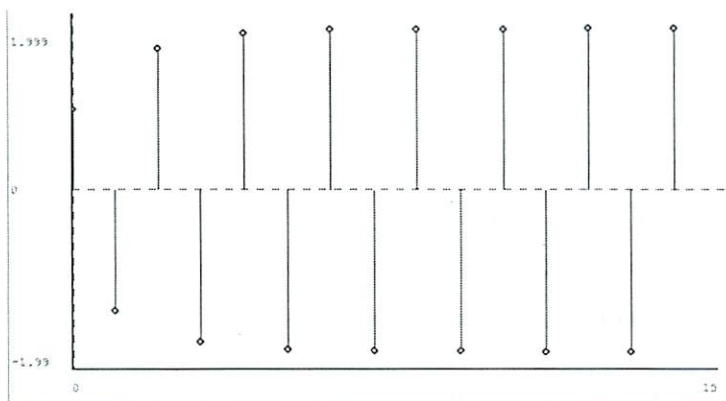
Poles:

Consider the following poles:

```
>>> H1.poles() = [-1.0, -0.5]
>>> H2.poles() = [(0.375+0.330j), (0.375-0.330j)]
>>> H3.poles() = [(0.05+0.234j), (0.05-0.234j)]
>>> H4.poles() = [1.2, -0.5]
```

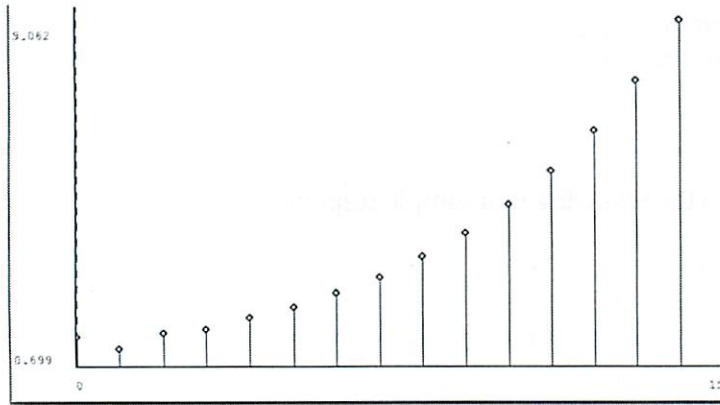
Part b. Deleted

Part c. Which (if any) of the poles lead to the following unit sample response?



Circle all correct answer(s): H_1 H_2 H_3 H_4 **none**

Part d. Which (if any) of the poles lead to the following unit sample response?



Circle all correct answers(s): H_1 H_2 H_3 H_4 **none**

Part e. Which (if any) of the poles lead to convergent unit-sample responses?

Circle all correct answers(s): H_1 H_2 H_3 H_4 **none**

Test Review

10/11
OH

Fall 09

- know largest pole is \ominus , alternating sign
 ≤ 1 bounded

$$(M_1)$$

back a pg

- Find the poles

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

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

so sub in

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

$$\frac{z^2}{z^2 - z + 0.5}$$

$$z^2 - z + 0.5$$

now quad formula

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

~~$$\frac{1 \pm \sqrt{1 - 4 \cdot 0.5 \cdot 1}}{2 \cdot 1}$$~~

$$\frac{1 \pm \sqrt{1 - 4 \cdot 0.5 \cdot 1}}{2 \cdot 1}$$

$$2 \cdot 1$$

$$= \frac{1 \pm \sqrt{-1}}{2}$$

$$= \frac{1}{2} \pm \frac{1}{2}j$$

if $\sqrt{-1}$
then $\sqrt{2}j$

?

it is called

j in CS/EE

$$M_2 = \frac{1}{1 - .4R - .05R^2}$$

$$= \frac{1}{1 - .4\left(\frac{1}{z}\right) - .05\left(\frac{1}{z}\right)^2} \cdot z^2$$

$$= \frac{z^2}{z^2 - .4z - .05}$$

$$z^2 - .4z - .05 \text{ set } = 0$$

$$= \frac{.4 \pm \sqrt{.16 - 4 \cdot 1 \cdot .05}}{2}$$

$$= \frac{.4 \pm \sqrt{.36}}{2}$$

$$= \frac{.4 \pm .6}{2}$$

$$= .2 \pm \frac{3}{10}$$

↳ then do it

$$.2 + .3 = .5$$

$$.2 - .3 = -.1$$

d) $|p_4|$ is it since > 1
+

- magnitude of largest pole

e) converge to unit-sample

- all are complex ones

$$|mag| < 1$$

$$p_2 + p_3$$

(if $mag > 1$ then would oscillate)

Spring 2010
d. COP
~~Q. 1~~

1 OOP (10 points)

The following definitions have been entered into the Python shell:

```
class F:  
    x = 0  
    y = 10  
    def __init__(self, y):  
        self.x = y  
    def a1(self, y):  
        self.x = max(self.x, y)  
        return self.x  
    def a2(self):  
        self.x = max(self.x, self.y)  
        return self.x  
class G(F):  
    def b(self):  
        self.y = self.x * self.x  
        return self.y
```

Write the values of the following expressions (enter None when there is no value; write Error when an error results and explain briefly why it's an error). Assume these expressions are evaluated one after the other (in each column).

f = F(5)

None

g = G()

Error: needs argument

f.a1(7)

7

g = G(3)

None

f.a1(3)

7

g.a1(5)

5

f.b()

Error: no such method

g.b()

25

f.a2()

10

g.a2()

25

2 Signals (20 points)

Given a signal x , we want to construct a signal y such that:

$$y[n] = a*x[n-1] + (1-a)*x[n-2]$$

for $0 \leq a \leq 1$.

Part a. Define a `Signal` subclass that represents this signal, given an instance of the `Signal` class representing x and the value of a . Construct the new signal, given a signal instance x , and a variable a , as follows:

```
y = MySig(x, a)
```

Do not use any of the existing `Signal` subclasses, such as `Rn` or `PolyR`.

```
class MySig(sig.Signal):
    def __init__(self, x, a):
        self.x = x
        self.a = a
    def sample(self, n):
        return self.a*self.x.sample(n-1) + (1-self.a)*self.x.sample(n-2)
```

Part b. Given a signal instance x , and a variable a , write an expression involving `sig.PolyR` that constructs an equivalent signal. To construct a polynomial use `poly.Polynomial`.

```
sig.PolyR(x, poly.Polynomial([1-a, a, 0]))
```

Part c. Write a Python procedure `settleTime` that is given:

- `s`: an instance of the `Signal` class;
- `m`: an integer, representing the max index to look at;
- `bounds`: a tuple of two values (`lo`, `hi`).

It returns an integer or `None`. The integer corresponds to the smallest sample index (`w`) such that the sample at index `w` and all the samples from `w` up to `m` are between `lo` and `hi` (less than or equal to `hi` and greater or equal to `lo`). It returns `None` if no such sample exists.

```
def settleTime(s, m, bounds):
    (lo,hi)=bounds
    def inBounds(x):
        return x>=lo and x<=hi
    w = 0
    for i in range(m):
        if not inBounds(s.sample(i)):
            w = i+1
    if w == m: return None
    else: return w
```

3 State Machines (20 points)

Write a state machine whose inputs are the characters of a string of “words” separated by spaces. For each input character, the machine should:

- output the string 'x' if the character is within a word, or
- if the input character is a space, then
 - output the most recent word if the most recent word is a number (all numerical digits), or
 - output None if the most recent input word is empty or contains any non-digits.

Assume there is never more than one consecutive space. For example:

```
>>> x = ' v1 11 1v '  
>>> FindNumbers().transduce(x)  
[None, 'x', 'x', None, 'x', 'x', '11', 'x', 'x', None]
```

Do not add any instance attributes to the state machines.

If `foo` is a single-character or multi-character string, the Python method `foo.isdigit()` returns True if there is at least one character in `foo` and all of the characters in `foo` are all digits.

```
class FindNumbers(sm.SM):  
    startState = ''  
    def getNextValues(self, state, inp):  
        if inp == ' ':  
            if state.isdigit():  
                return ('', state)  
            else:  
                return ('', None)  
        else:  
            return (state+inp, 'x')
```

4 Difference Equations (10 points)

Newton's law of cooling states that:

The change in an object's temperature from one time step to the next is proportional to the difference (on the earlier step) between the temperature of the object and the temperature of the environment, as well as to the length of the time step.

Let

- $o[n]$ be temperature of object
- $s[n]$ be temperature of environment
- T be the duration of a time step
- K be the constant of proportionality

Part a. Write a difference equation for Newton's law of cooling. Be sure the signs are such that the temperature of the object will eventually equilibrate with that of the environment.

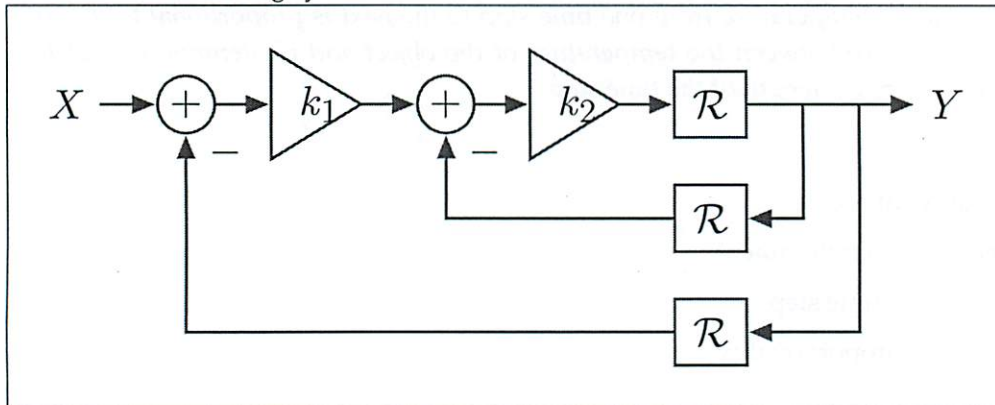
$$o[n] = o[n - 1] + TK(s[n - 1] - o[n - 1])$$

Part b. Write the system function corresponding to this equation (show your work):

$$H = \frac{O}{S} = \frac{KTR}{1 - (1 - KT)R}$$

5 Signals and Systems (20 points)

Consider the following system:



Part a. Write the system function:

$$H = \frac{Y}{X} = \frac{k_1 k_2 R}{1 + k_2 R^2 (1 + k_1)}$$

Part b.

Let $k_1 = 1$ and $k_2 = -2$. Assume that the system starts "at rest" (all signals are zero) and that the input signal X is the unit sample signal. Determine $y[0]$ through $y[3]$.

$$y[0] = \boxed{0}$$

$$y[1] = \boxed{-2}$$

$$y[2] = \boxed{0}$$

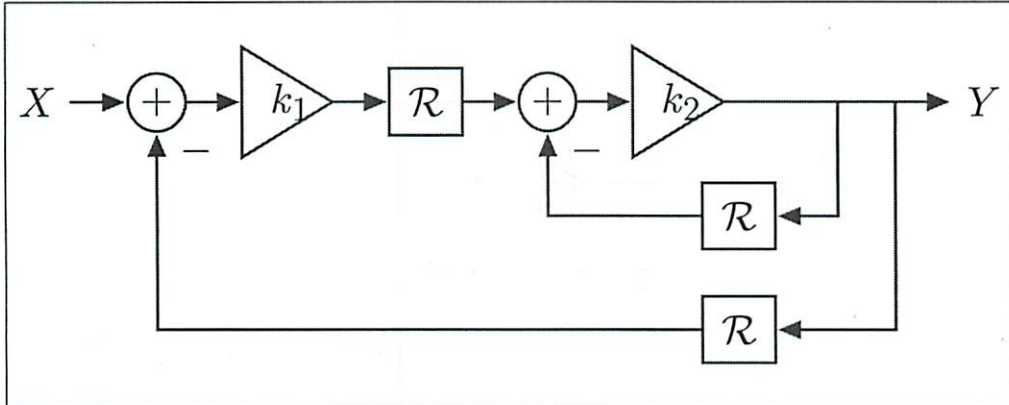
$$y[3] = \boxed{-8}$$

Part c. Let $k_1 = 1$ and $k_2 = -2$, determine the poles of H .

Enter poles or **none** if there are no poles:

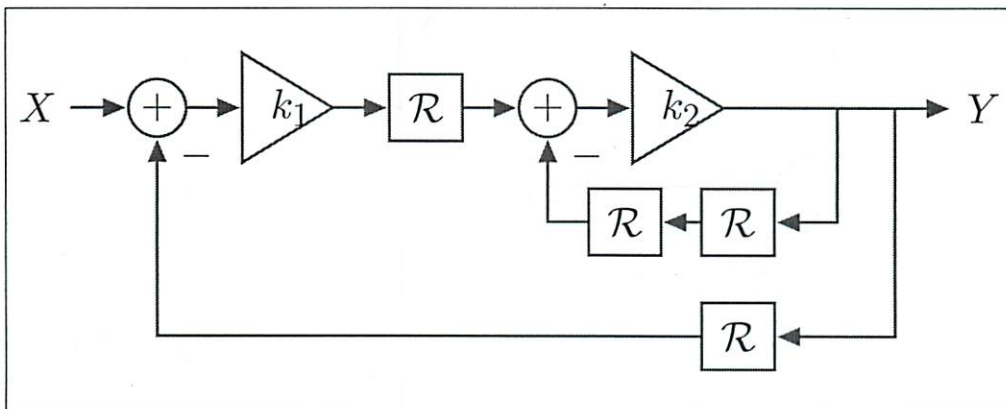
$\boxed{2, -2}$

Part d. For each of the systems below indicate whether the system is equivalent to this one: (Remember that there can be multiple “equivalent” representations for a system.) If you write clearly the system function for these systems, we may be able to give you partial credit.



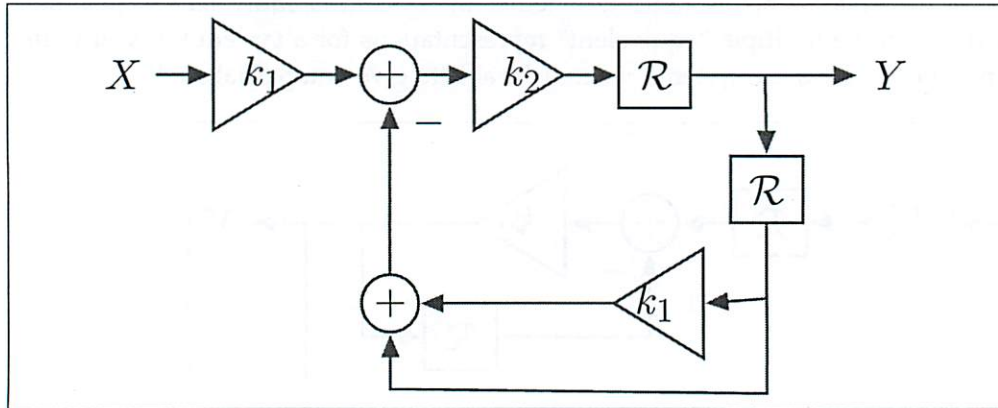
Equivalent to H (yes/no)?

No



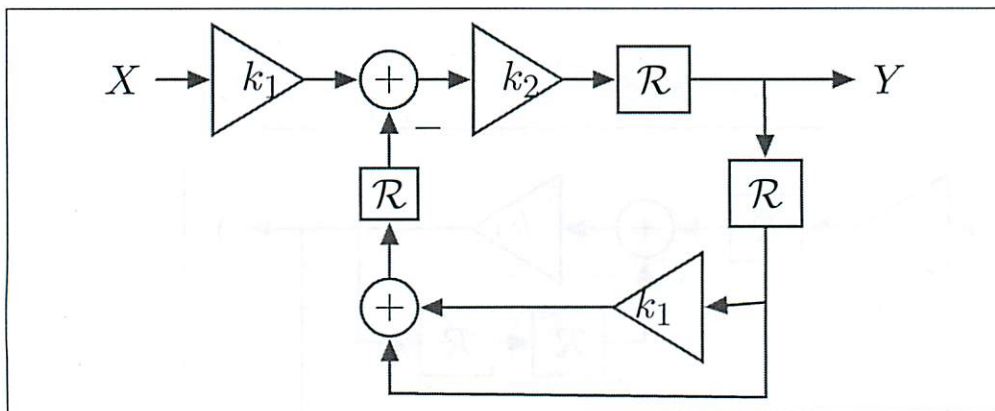
Equivalent to H (yes/no)?

Yes



Equivalent to H (yes/no)?

Yes



Equivalent to H (yes/no)?

No

6 System Behaviors (20 points)

Part a. Find the poles for the following system functions:

$$H_1(\mathcal{R}) = \frac{1}{1 - \mathcal{R} + 0.5\mathcal{R}^2}$$

Poles:

$$0.5 \pm 0.5j$$

$$H_2(\mathcal{R}) = \frac{1}{1 - 0.4\mathcal{R} - 0.05\mathcal{R}^2}$$

Poles:

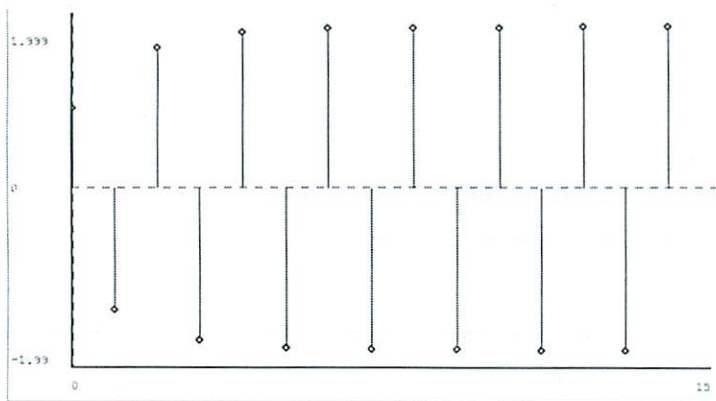
$$0.5, -0.1$$

Consider the following poles:

```
>>> H1.poles() = [-1.0, -0.5]
>>> H2.poles() = [(0.375+0.330j), (0.375-0.330j)]
>>> H3.poles() = [(0.05+0.234j), (0.05-0.234j)]
>>> H4.poles() = [1.2, -0.5]
```

Part b. Deleted

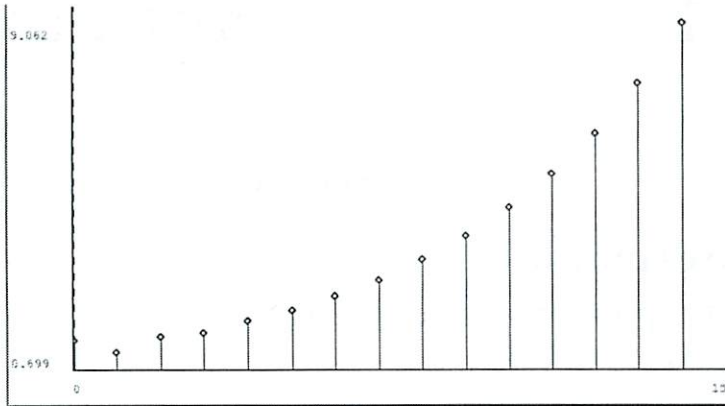
Part c. Which (if any) of the poles lead to the following unit sample response?



Circle all correct answers(s):

 H_1
 H_2
 H_3
 H_4
 none

Part d. Which (if any) of the poles lead to the following unit sample response?



Circle all correct answers(s): H_1 H_2 H_3 H_4 none

Part e. Which (if any) of the poles lead to convergent unit-sample responses?

Circle all correct answers(s): H_1 H_2 H_3 H_4 none

6.01 Midterm 1 Solutions: Spring 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.	/10
2.	/20
3.	/15
4.	/8
5.	/10
6.	/12
7.	/25
total:	/100

6.01 Midterm 1 : Spring 2010

Name:	Section:
--------------	-----------------

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.	/10
2.	/20
3.	/15
4.	/8
5.	/10
6.	/12
7.	/25
total:	/100

1 OOP (10 points)

The following definitions have been entered into a Python shell:

```

class A:
    yours = 0
    def __init__(self, inp):
        self.yours = inp
    def give(self, amt):
        self.yours = self.yours + amt
        return self.yours
    def howmuch(self):
        return (A.yours, self.yours)

```

```

class B(A):
    yours = 0
    def give(self, amt):
        B.yours = B.yours + amt
        return A.howmuch(self)
    def take(self, amt):
        self.yours = self.yours - amt
        return self.yours
    def howmuch(self):
        return (B.yours, A.yours, self.yours)

```

ecorrect? - ok wenn A: r says which to use

write the output!

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).

self. yours = 5

test = A(5)

test = B(5)
 B.yours = 5

test.take(2)

test.take(2) *carriers over remember init was*
 ** yours gets reset from init!*

test.give(6)

test.give(6)

test.howmuch()

test.howmuch()

'can you do that? A. refers to all instances of class'
 or (0, 11) ✓

(6, 0, 3) test on PC to play w/

OOP exploration

10/7

- can have B.years

- global among all instances

can call it directly

print B.years

When you reassign a name to a new object

it does NOT carry over

init gets inherited

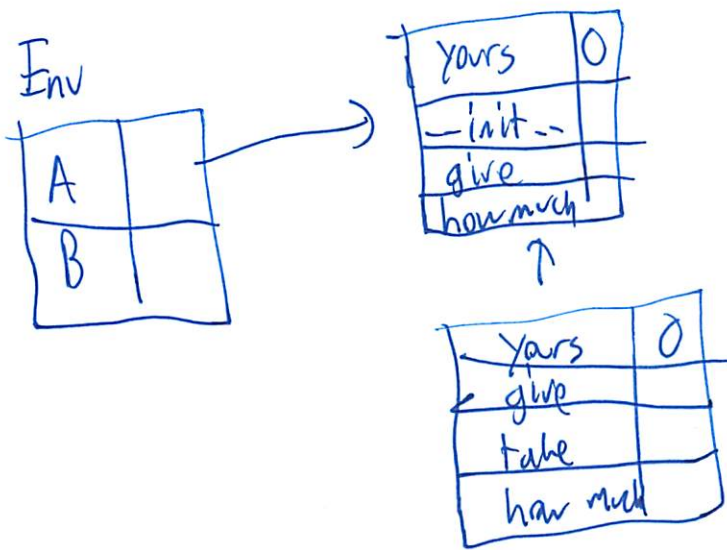
Q10

w/ll
OH

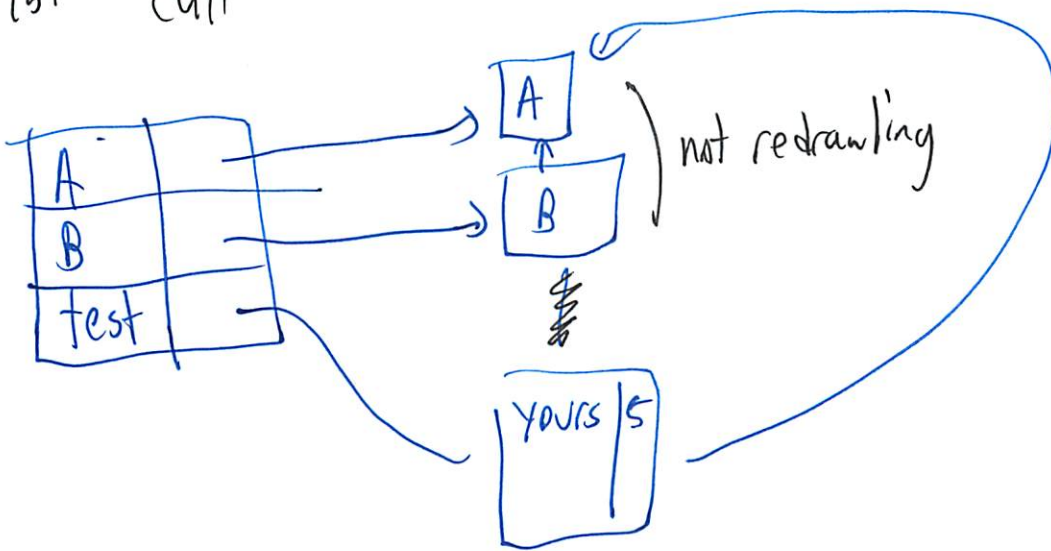
LOOP

Correct way to do this is w/ diagrams

- kinda did in my head
- will be a lot of diagrams to draw
- w/ env



1st call



2 The Best and the Brightest (20 points)

Here are some class definitions, meant to represent a collection of students making up the student body of a prestigious university.

```
class Student:
    def __init__(self, name, iq, salary, height):
        self.name = name
        self.iq = iq
        self.salary = salary
        self.height = height
```

```
class StudentBody:
    def __init__(self):
        self.students = []
    def addStudent(self, student):
        self.students.append(student)
    def nameOfSmartest(self):
        # code will go here
    def funOfBest(self, fun, feature):
        # code will go here
```

2 separate classes

The StudentBody class is missing the code for two methods:

- `nameOfSmartest`: returns the name attribute of the student with the highest IQ
- `funOfBest`: takes a procedure fun and a procedure feature as input, and returns the result of applying procedure fun to the student for whom the procedure feature yields the highest value

For the first two problems below, assume that these methods have been implemented.

Here is a student body:

```
jody = Student('Jody', 100, 100000, 80)
chris = Student('Chris', 150, 40000, 62)
dana = Student('Dana', 120, 2000, 70)
aardvarkU = StudentBody()
aardvarkU.addStudent(jody)
aardvarkU.addStudent(chris)
aardvarkU.addStudent(dana)
```

1. What is the result of evaluating `aardvarkU.nameOfSmartest()`?

`'Chris'` ✓

2. Write a Python expression that will compute the name of the person who has the greatest value of IQ + height in `aardvarkU` (not just for the example student body above). You can define additional procedures if you need them.

```

best = [0, 0]
for x in aardvarkU.students:
    if (x.height + x.IQ) > best[1]:
        best[0] = x.name
        best[1] = x.height + x.IQ
return best[0]

```

they did w/ argmax

can do for loop and have temp - or some sort of sorting

3. Implement the `nameOfSmartest` method. For full credit, use `util.argmax` (defined below) or the `funOfBest` method. *only* - see docs

If `l` is a list of items and `f` is a procedure that maps an item into a numeric score, then `util.argmax(l, f)` returns the element of `l` that has the highest score.

```

return util.argmax(aardvarkU.students, proc height + IQ, name)

```

don't know what format it wants

lambda x: x.IQ

what had would have been lambda x: x.iq + x.height

- loop is hackish - but I am more comfortable

4. Implement the `funOfBest` method. For full credit, use `util.argmax`.

```
return util.argmax(self.students, feature).fun
```

I did not do it w/ implementing a method
- need to see if they just want you to return
- but they might have accepted this

Repeated from the start of the problem:

```
class Student:
    def __init__(self, name, iq, salary, height):
        self.name = name
        self.iq = iq
        self.salary = salary
        self.height = height
```

```
class StudentBody:
    def __init__(self):
        self.students = []
    def addStudent(self, student):
        self.students.append(student)
    def deleteStudent(self, student):
        self.students.remove(student)
    def nameOfSmartest(self):
        pass
    def funOfBest(self, fun, feature):
        pass
```

Here is a student body:

```
jody = Student('Jody', 100, 100000, 80)
chris = Student('Chris', 150, 40000, 62)
dana = Student('Dana', 120, 2000, 70)
aardvarkU = StudentBody()
aardvarkU.addStudent(jody)
aardvarkU.addStudent(chris)
aardvarkU.addStudent(dana)
```

My issue is always
Python minute

3 SM to DE (15 Points)

Here is the definition of a class of state machines:

```
class Thing(SM):
    startState = [0, 0, 0, 0]
    def getNextValues(self, state, inp):
        result = state[0] * 2 + state[2] * 3
        newState = [state[1], result, state[3], inp]
        return (newState, result)
```

Should ~~not~~ be able
to fix this

1. What is the result of evaluating

```
Thing().transduce([1, 2, 0, 1, 3])
```

$[0, 0, 3, 6, 6]$ ✓
? list right?

2. The state machine above describes the behavior of an LTI system starting at rest. Write a difference equation that describes the same system as the state machine. *taugh one*

$y[n] = 2y[n-2] + 3x[n-2]$ ✓ Bings

inp state

1 $[0, 0, 0, 0]$
result = 0 = output
new state $[0, 0, 0, 1]$ *read carefully*

2 $[0, 0, 0, 1]$
result = 0
 $[0, 0, 1, 2]$

0 $[0, 0, 1, 2]$
result = 3
 $[0, 3, 2, 0]$
 $[0, 3, 2, 0]$
6
 $[3, 6, 0, 4]$

3 $[3, 6, 0, 1]$
6 + 0
 $[6, 6, 1, 3]$
→ $y[n] = 2y[n-2] + 3x[n-2]$
hope that was you were
supposed to do

4 Diagnosis (8 points)

What, if anything, is wrong with each of the following state machine definitions? The syntax is correct (that is, they are all legal Python programs), so that is not a problem. Write a phrase or a single sentence if something is wrong or "nothing" if nothing is wrong.

1.

```
class M1(SM):
    startState = 0
    def getNextValues(self, state, inp):
        return inp + state
```

Only returning 1 parameter → state
No output ✓

2.

```
class M2(SM):
    startState = 0
    def getNextValues(self, state, inp):
        return (inp+state, (state, state))
```

↑ returning a duplicate tuple
Can it be added? - No tuples can't be changed

⊗

is fine

3.

```
class M3(SM):
    def __init__(self):
        self.startState = [1, 2, 3]
    def getNextValues(self, state, inp):
        state.append(inp)
        result = sum(state) / float(len(state))
        return (result, state)
```

flipped
~~It would work, but not as expected.~~
~~They were looking that~~

4.

```
class M4(SM):
    startState = -1
    def __init__(self, v):
        self.value = v
    def getNextValues(self, state, inp):
        if state > inp:
            self.value = self.value + state
        result = self.value * 2
        return (state, result)
```

breaks SM logic

- modifies something in getNextValues
 which is not what I was

thinking

- they might not accept mine
 as specific enough

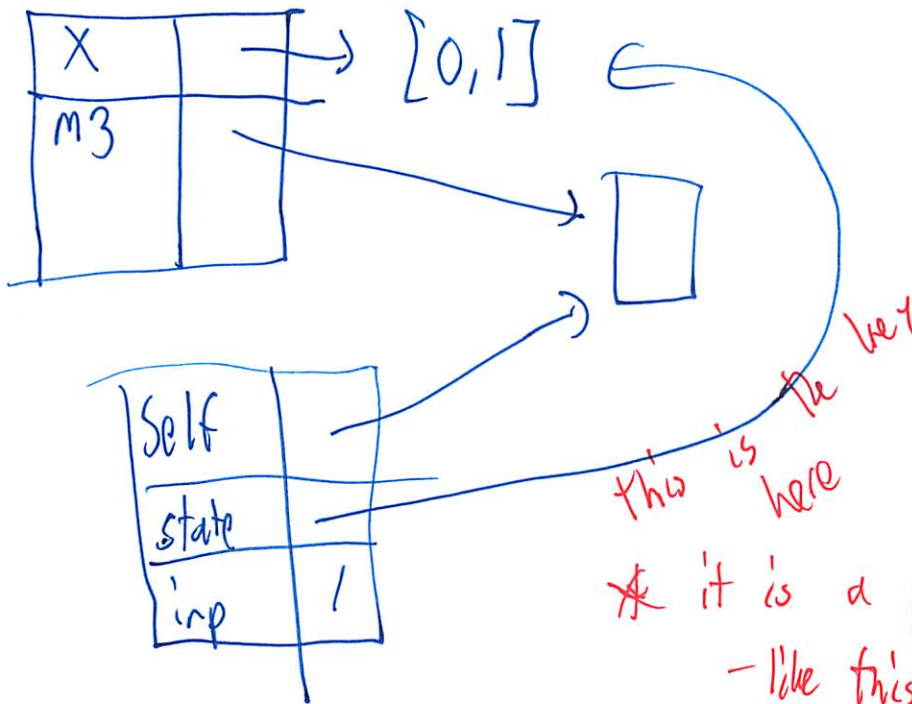
Spring 2010

lolll
OH

- Must be pure function
 - nothing else in system changes
 - safe to call

3. ~~How~~ But state is ~~is~~ inside it

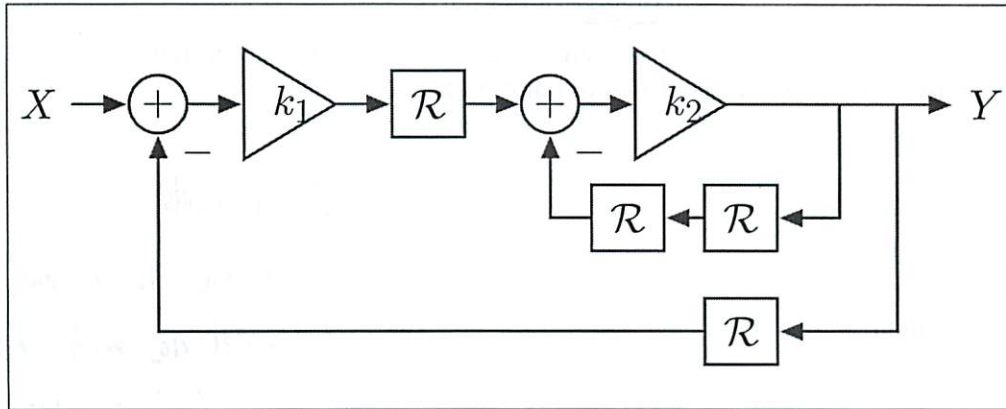
Oh ~~m~~ x is actually changed
- when you draw the diagram



* it is a pointer, not a copy *
- like this in most languages

5 Picture to Machine (10 Points)

Use `sm.Gain`, `sm.Delay`, `sm.FeedbackAdd`, `sm.FeedbackSubtract`, and `sm.Cascade` to make a state machine that is equivalent to the following system:



Assume that the system starts at rest, that is, all the signals are zero at time zero. Assume also that the variables `k1` and `k2` are already defined.

Recall that `sm.Gain` takes a gain as an argument, `sm.Delay` takes an initial output value as an argument and each of `sm.Cascade`, `sm.FeedbackAdd`, and `sm.FeedbackSubtract`, takes two state machines as arguments.

Use indentation to highlight the structure of your answer.

$$y = \text{FeedbackSubtract}(b, R)$$

$$b = \text{Cascade}(k_1, \text{Cascade}(R, a))$$

$$a = \text{FeedbackSubtract}(\text{Gain}(k_2), \text{Cascade}(R, R))$$

on test would rewrite, but don't really see what they mean by indent

6 On the Verge (12 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,
- 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

1.

$$10y[n] - y[n-1] = 8x[n-3]$$

diverge? Yes or No

positive/alternate/oscillate

find roots

- need to memorize

- well no, will have

lecture notes,

2.

$$y[n] = -y[n-1] - 10y[n-2] + x[n]$$

diverge? Yes or No

positive/alternate/oscillate

$$10Y - YR = 8XR^3$$

$$Y(10-R) = 8XR^3$$

$$\frac{Y}{X} = \frac{8R^3}{(10-R)}$$

← how supposed to find roots of w/o calc?

- well what are roots

- root of denom

$$10 - R = 0$$

$$R = 10$$

$$\textcircled{10}$$

$$\textcircled{+}$$

change

no

not

$$Y + YR + 10YR^2 = X$$

$$\frac{Y}{X} = \frac{1}{(10R^2 + R + 1)}$$

$$10R^2 + R = -1$$

$$R(10R + 1) = -1$$

- need to review root finding by hand!

- quadratic eq!

~~Best + Bright~~

$$10R^2 + R + 1$$

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

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

$$\frac{-1 \pm \sqrt{1^2 - 4(10)(1)}}{2 \cdot 10}$$

$$\frac{-1 \pm \sqrt{-39}}{40}$$

negative

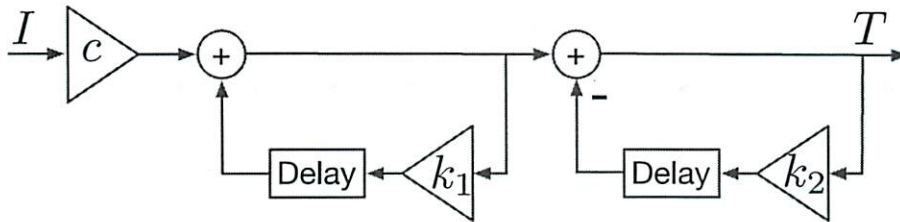
imaginary
how we solve that

- all you need to know

7 What's Cooking? (25 Points)

Sous vide cooking involves cooking food at a very precise, fixed temperature T (typically, low enough to keep it moist, but high enough to kill any pathogens). In this problem, we model the behavior of the heater and water bath used for such cooking. Let I be the current going into the heater, and c be the proportionality constant such that Ic is the rate of heat input.

The system is thus described by the following diagram:



1. a. Give the system function: $H = \frac{T}{I} =$

$$\frac{T}{I} = c + k_1 c R + k_2 c R^2$$

T is output
 they use T

- b. Give a difference equation for the system: $T[n] =$

$$T[n] = I[n] \cdot c + k_1 c T[n-1] + k_2 c (T[n-2])$$

$(k_1 - k_2)T[n-1] + k_1 k_2 c T[n-2]$

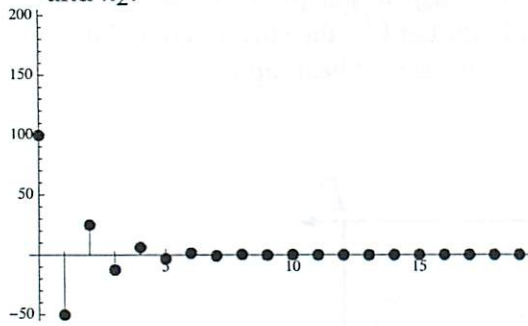
from 2 terms added together
 from last going around
 around both

$$Y = cX + k_1 c X R + k_2 c X R^2$$

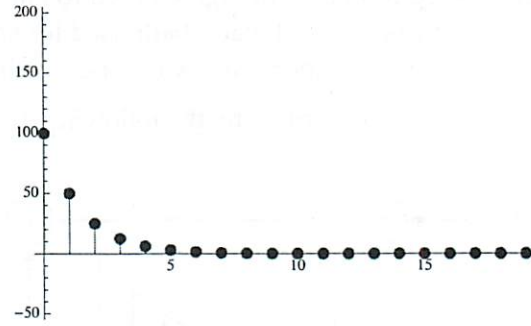
ok ~~the~~ 4 times

why did I screw this up so bad

2. Let the system start at rest (all signals are zero). Suppose $I[0] = 100$ and $I[n] = 0$ for $n > 0$. Here are plots of $T[n]$ as a function of n for this system for $c = 1$ and different values of k_1 and k_2 .



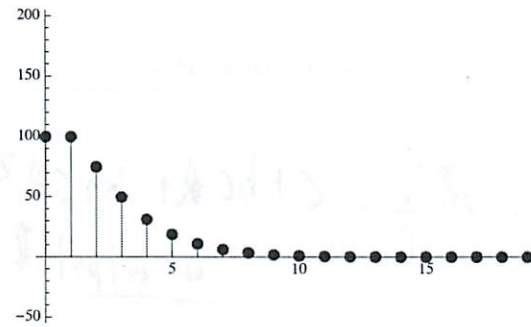
a



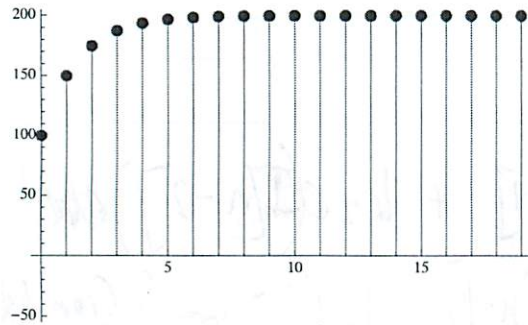
b



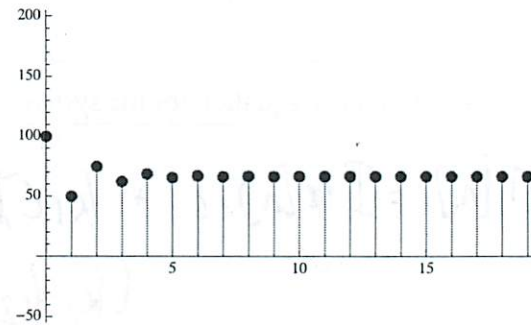
c



d



e



f

a. Which of the plots above corresponds to $k_1 = 0.5$ and $k_2 = 0$?

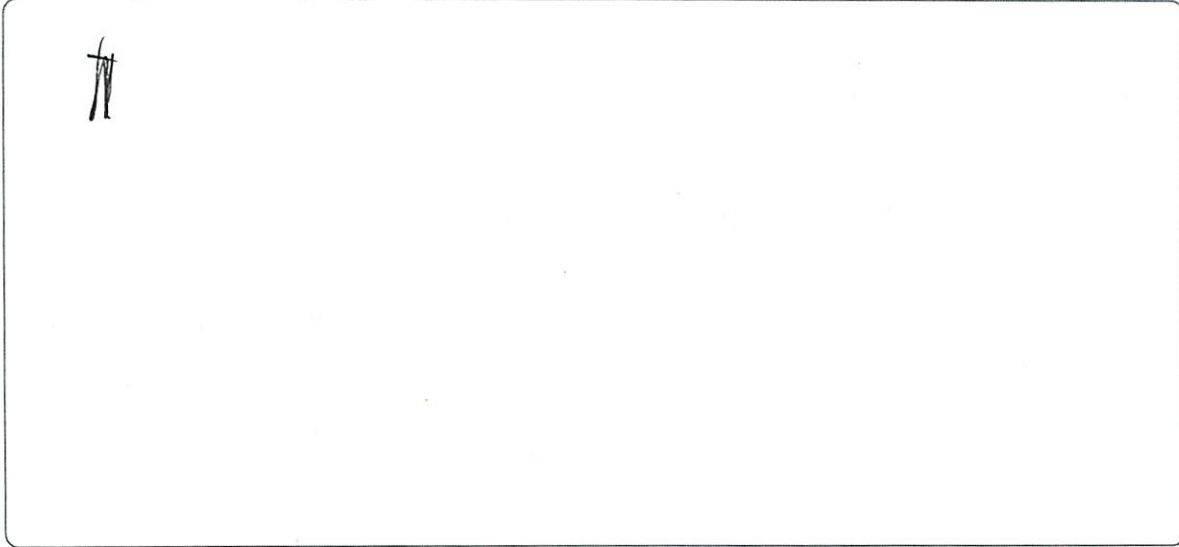
Circle all correct answers: a b c d e f none

b. Which of the plots above corresponds to $k_1 = 1$ and $k_2 = 0.5$?

Circle all correct answers: a b c d e f none

how to test
 - can't like last time - or find poles
 do both to (D)

3. Let $k_1 = 0.5$, $k_2 = 3$, and $c = 1$. Determine the poles of H , or **none** if there are no poles.



$$T[n] = (0.5 - 3) T[n-1] + 3 \cdot 0.5 T[n-2] + c I[n]$$

$$y = 2.5 z^{-1} + 1.5 z^{-2} + x$$

$$\frac{Y}{X} = \frac{1}{(1 - 2.5z^{-1} + 1.5z^{-2})}$$

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

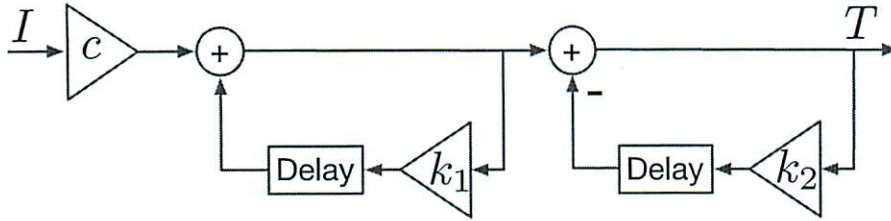
$$\frac{2.5 \pm \sqrt{2.5^2 - 4 \cdot 1 \cdot 1.5}}{2(1.5)}$$

$$\frac{2.5 \pm \sqrt{4.25 - 6}}{3}$$

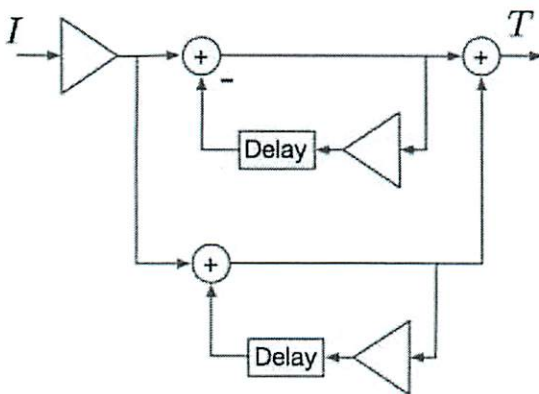
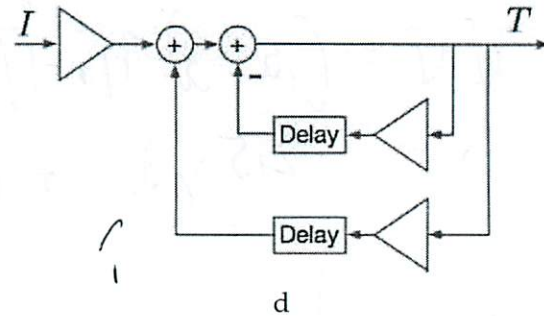
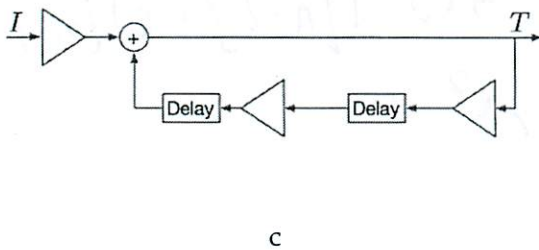
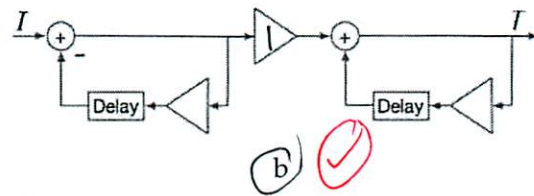
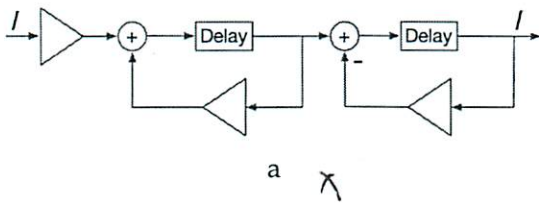
Complex
need calc

5
3) make sure to
bring calc

4. Here is the original system:

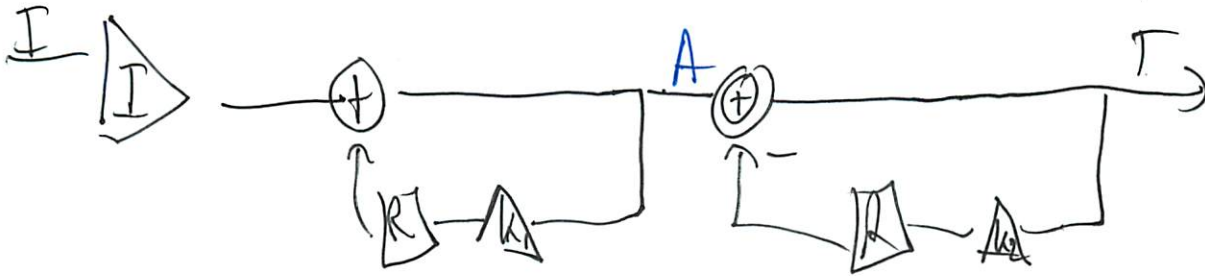


Circle all of the following systems that are equivalent (for some setting of the gains in the triangles) to the original system.



could be
same poles

could calculate R for each



Systematically turn into expression
* start backwards

$$T[n] = \text{---}$$

unravel like a parser

$$T = A - R k_2 \cdot T$$

$$A = C I + R k_1 \cdot A$$

his way
2 formulas to help

Cascade

$$H = H_1 H_2$$

$$\frac{N_1 N_2}{D_1 D_2}$$

Feedback subtract (Black's)

$$t_l = \frac{t_1}{1 + H_1 H_2}$$

② Or do it this way (from start)

2nd part

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

$$\uparrow H_2 \Rightarrow Y = R \cdot k_2 X$$
$$\frac{Y}{X} = \frac{Rk_2}{1}$$

$$\frac{1}{1 + \frac{1}{1} \cdot \frac{Rk_2}{1}}$$

$$\frac{I}{A} = \frac{1}{1 + Rk_2}$$

1st part

$$\frac{A}{I} = 1$$

$$H_1 = 1$$

$$H_2 = -Rk_1$$

③

$$C \cdot \frac{1}{1 - Rk_1}$$

Together

$$C \cdot \frac{1}{1 - Rk_1} \cdot \frac{1}{1 + Rk_2}$$

- much simpler that way

- think of it that way instead

- same for that other practice problem I did

b Just multiply out + convert

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

∴ can do

↓ Fall 09 #5 in similar manner

4

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

then make table

x	0	0	1	0	0	0		
y	0	0	0	-2	0	-8		
n	-2	-1	0	1	2	3	4	5

? have all that before you start
then run through in blue

Spring 2010 Solutions

1 OOP (10 points)

The following definitions have been entered into a Python shell:

```
class A:
    yours = 0
    def __init__(self, inp):
        self.yours = inp
    def give(self, amt):
        self.yours = self.yours + amt
        return self.yours
    def howmuch(self):
        return (A.yours, self.yours)

class B(A):
    yours = 0
    def give(self, amt):
        B.yours = B.yours + amt
        return A.howmuch(self)
    def take(self, amt):
        self.yours = self.yours - amt
        return self.yours
    def howmuch(self):
        return (B.yours, A.yours, self.yours)
```

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).

test = A(5)

None

test.take(2)

Error

test.give(6)

11

test.howmuch()

(0, 11)

test = B(5)

None

test.take(2)

3

test.give(6)

(0, 3)

test.howmuch()

(6, 0, 3)

2 The Best and the Brightest (20 points)

Here are some class definitions, meant to represent a collection of students making up the student body of a prestigious university.

```
class Student:
    def __init__(self, name, iq, salary, height):
        self.name = name
        self.iq = iq
        self.salary = salary
        self.height = height

class StudentBody:
    def __init__(self):
        self.students = []
    def addStudent(self, student):
        self.students.append(student)
    def nameOfSmartest(self):
        # code will go here
    def funOfBest(self, fun, feature):
        # code will go here
```

The StudentBody class is missing the code for two methods:

- `nameOfSmartest`: returns the name attribute of the student with the highest IQ
- `funOfBest`: takes a procedure `fun` and a procedure `feature` as input, and returns the result of applying procedure `fun` to the student for whom the procedure `feature` yields the highest value

For the first two problems below, assume that these methods have been implemented.

Here is a student body:

```
jody = Student('Jody', 100, 100000, 80)
chris = Student('Chris', 150, 40000, 62)
dana = Student('Dana', 120, 2000, 70)
aardvarkU = StudentBody()
aardvarkU.addStudent(jody)
aardvarkU.addStudent(chris)
aardvarkU.addStudent(dana)
```

1. What is the result of evaluating `aardvarkU.nameOfSmartest()`?

```
'Chris'
```

2. Write a Python expression that will compute the name of the person who has the greatest value of IQ + height in `aardvarkU` (not just for the example student body above). You can define additional procedures if you need them.

```
aardvarkU.funOfBest(lambda x: x.name, lambda x: x.iq + x.height)
```

3. Implement the `nameOfSmartest` method. For full credit, use `util.argmax` (defined below) or the `funOfBest` method.

If `l` is a list of items and `f` is a procedure that maps an item into a numeric score, then `util.argmax(l, f)` returns the element of `l` that has the highest score.

```
def nameOfSmartest(self):  
    return util.argmax(self.students, lambda x: x.iq).name  
  
# or  
  
def nameOfSmartest(self):  
    return funOfBest(lambda x: x.name, lambda x: x.iq)
```


4. Implement the `funOfBest` method. For full credit, use `util.argmax`.

```
def funOfBest(self, fun, feature):  
    return fun(util.argmax(self.students, feature))
```

Repeated from the start of the problem:

```
class Student:  
    def __init__(self, name, iq, salary, height):  
        self.name = name  
        self.iq = iq  
        self.salary = salary  
        self.height = height
```

```
class StudentBody:  
    def __init__(self):  
        self.students = []  
    def addStudent(self, student):  
        self.students.append(student)  
    def deleteStudent(self, student):  
        self.students.remove(student)  
    def nameOfSmartest(self):  
        pass  
    def funOfBest(self, fun, feature):  
        pass
```

Here is a student body:

```
jody = Student('Jody', 100, 100000, 80)  
chris = Student('Chris', 150, 40000, 62)  
dana = Student('Dana', 120, 2000, 70)  
aardvarkU = StudentBody()  
aardvarkU.addStudent(jody)  
aardvarkU.addStudent(chris)  
aardvarkU.addStudent(dana)
```

3 SM to DE (15 Points)

Here is the definition of a class of state machines:

```
class Thing(SM):
    startState = [0, 0, 0, 0]
    def getNextValues(self, state, inp):
        result = state[0] * 2 + state[2] * 3
        newState = [state[1], result, state[3], inp]
        return (newState, result)
```

1. What is the result of evaluating

```
Thing().transduce([1, 2, 0, 1, 3])
```

```
[0, 0, 3, 6, 6]
```

2. The state machine above describes the behavior of an LTI system starting at rest. Write a difference equation that describes the same system as the state machine.

```
 $y[n] = 2 * y[n-2] + 3 * x[n - 2]$ 
```

4 Diagnosis (8 points)

What, if anything, is wrong with each of the following state machine definitions? The syntax is correct (that is, they are all legal Python programs), so that is not a problem. Write a phrase or a single sentence if something is wrong or “nothing” if nothing is wrong.

1.

```
class M1(SM):
    startState = 0
    def getNextValues(self, state, inp):
        return inp + state
```

Should return tuple of new state and output

2.

```
class M2(SM):
    startState = 0
    def getNextValues(self, state, inp):
        return (inp+state, (state, state))
```

This is fine.

3.

```
class M3(SM):
    def __init__(self):
        self.startState = [1, 2, 3]
    def getNextValues(self, state, inp):
        state.append(inp)
        result = sum(state) / float(len(state))
        return (result, state)
```

This modifies the state argument, by doing `state.append(inp)`

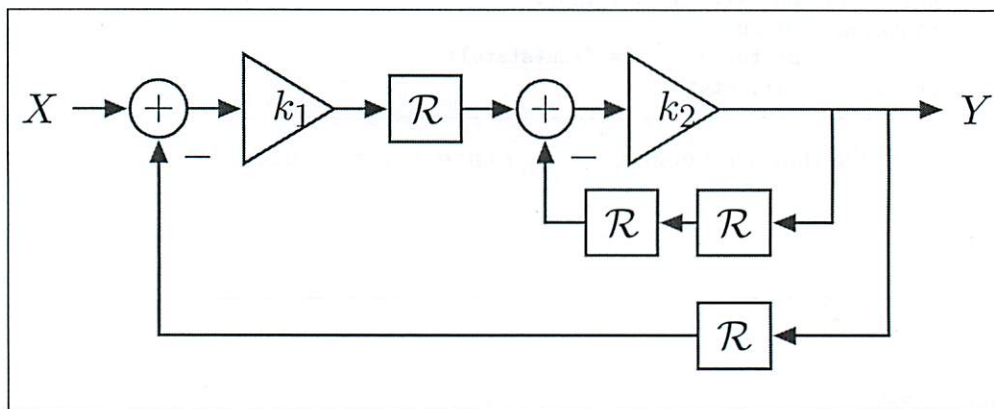
4.

```
class M4(SM):
    startState = -1
    def __init__(self, v):
        self.value = v
    def getNextValues(self, state, inp):
        if state > inp:
            self.value = self.value + state
        result = self.value * 2
        return (state, result)
```

This modifies the attribute value in `getNextValues`.

5 Picture to Machine (10 Points)

Use `sm.Gain`, `sm.Delay`, `sm.FeedbackAdd`, `sm.FeedbackSubtract`, and `sm.Cascade` to make a state machine that is equivalent to the following system:



Assume that the system starts at rest, that is, all the signals are zero at time zero. Assume also that the variables k_1 and k_2 are already defined.

Recall that `sm.Gain` takes a gain as an argument, `sm.Delay` takes an initial output value as an argument and each of `sm.Cascade`, `sm.FeedbackAdd`, and `sm.FeedbackSubtract`, takes two state machines as arguments.

Use indentation to highlight the structure of your answer.

```
FeedbackSubtract(Cascade(Cascade(Gain(k1), R()),
                          FeedbackSubtract(Gain(k2),
                                          Cascade(R(), R()))),
                R())
```

6 On the Verge (12 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,
- 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

1.

$$10y[n] - y[n - 1] = 8x[n - 3]$$

diverge? Yes or No

No

positive/alternate/oscillate

Positive

2.

$$y[n] = -y[n - 1] - 10y[n - 2] + x[n]$$

diverge? Yes or No

Yes

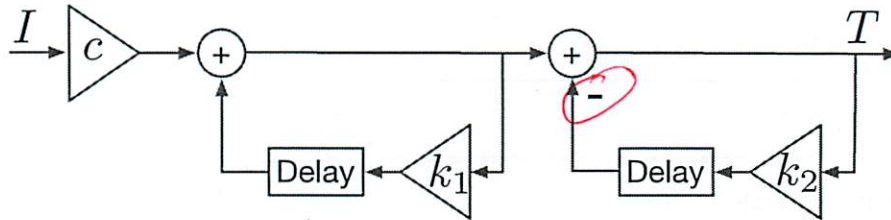
positive/alternate/oscillate

Oscillates, pole is complex

7 What's Cooking? (25 Points)

Sous vide cooking involves cooking food at a very precise, fixed temperature T (typically, low enough to keep it moist, but high enough to kill any pathogens). In this problem, we model the behavior of the heater and water bath used for such cooking. Let I be the current going into the heater, and c be the proportionality constant such that Ic is the rate of heat input.

The system is thus described by the following diagram:



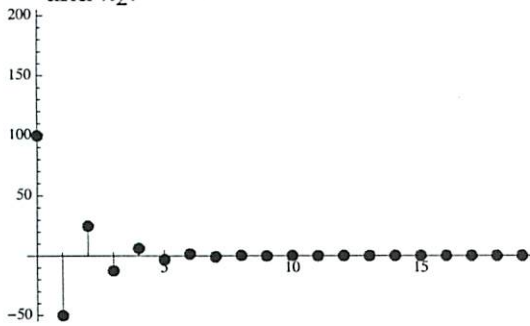
1. a. Give the system function: $H = \frac{T}{I} =$

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

- b. Give a difference equation for the system: $T[n] =$

$$T[n] = (k_1 - k_2)T[n - 1] + k_1 k_2 T[n - 2] + cI[n]$$

2. Let the system start at rest (all signals are zero). Suppose $I[0] = 100$ and $I[n] = 0$ for $n > 0$. Here are plots of $T[n]$ as a function of n for this system for $c = 1$ and different values of k_1 and k_2 .



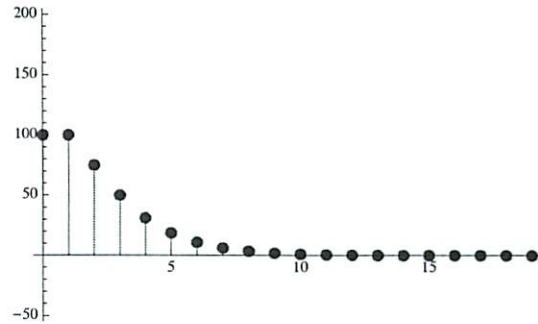
a



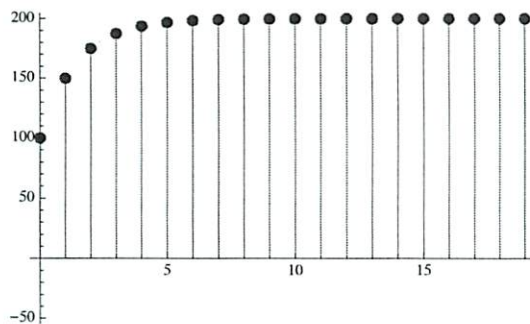
b



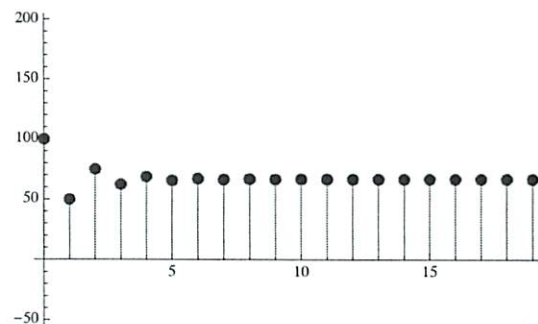
c



d



e



f

a. Which of the plots above corresponds to $k_1 = 0.5$ and $k_2 = 0$?

Circle all correct answers: a b c d e f none

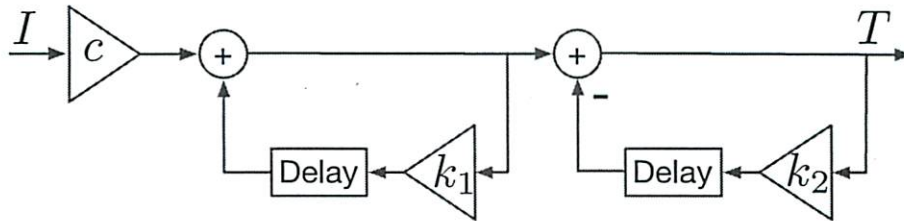
b. Which of the plots above corresponds to $k_1 = 1$ and $k_2 = 0.5$?

Circle all correct answers: a b c d e f none

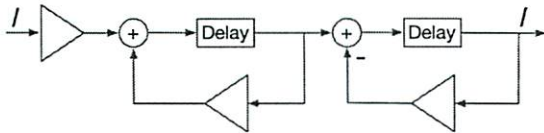
3. Let $k_1 = 0.5$, $k_2 = 3$, and $c = 1$. Determine the poles of H , or **none** if there are no poles.

Poles at 0.5 and -3

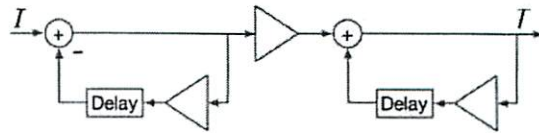
4. Here is the original system:



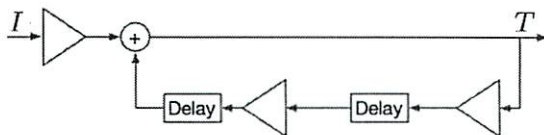
Circle all of the following systems that are equivalent (for some setting of the gains in the triangles) to the original system.



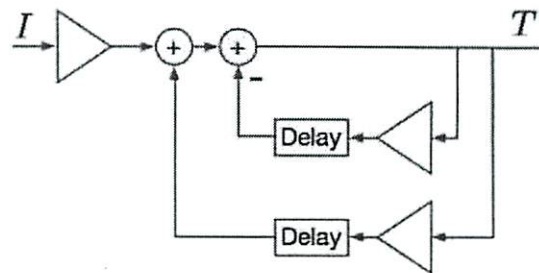
a



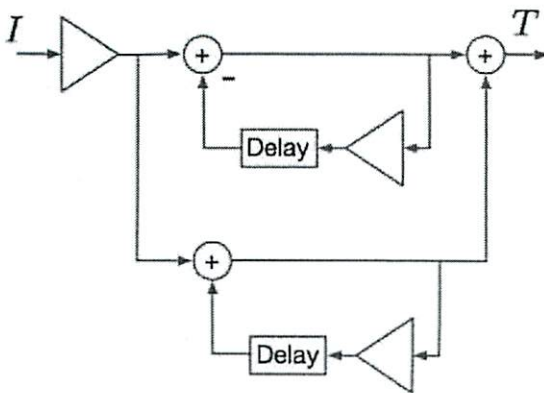
b



c



d



e

System b has the same system function. System e has the same poles.

We required b to be circled, and considered e to be correct whether circled or not.