MASSACHVSETTS INSTITVTE OF TECHNOLOGY
Department of Electrical Engineering and Computer Science
6.01—Introduction to EECS I
Spring Semester, 2008

Assignment 5, Issued: Tuesday, Mar. 4

Overview of this week's work

In software lab

- Work through the software lab.
- Paste your solution to Question 7 into the tutor.

Before the start of your design lab on Mar 4 or 5

- Read the class notes and review the lecture handout.
- Do the on-line tutor problems in section PS.6.2.
- Read the entire description of the design lab, so that you will be ready to work on it when you get to lab.

In design lab

- Take the nanoquiz in the first 15 minutes; don't be late.
- Work through the design lab with a partner, and take good notes on the results of your work.

At the beginning of your next software lab on Mar 11 or 12

- Submit online tutor problems in section PS.6.3.
- Submit written solutions to questions 7–9, 11, 14–19, and 20. All written work must conform to the homework guidelines on the web page.

Get the lab5 files via `athrun 6.01 update` or from the home page.

Software Lab: System Functions for Robot Controller

We saw in lecture this week that the behavior of a linear time-independent system can be completely characterized using a ratio of two polynomials in R, the delay operator. We call this characterization the *system function*. Further, we saw that systems composed of multiple LTI components can be described using system functions that are relatively simple compositions of the system functions of the components. It is possible to solve such problems by hand, with a paper and pencil, but why do that when we can get a computer to do the work for us?!

In this software lab and the next we will build and use a set of tools for analyzing, simulating, and combining system functions. The file `SystemFunctionSkeleton.py` contains the definition of the class `SystemFunction`, with some methods filled in, and others just documented, but with `pass` instead of the method body. We'll fill in its methods this week and next. We'll also give you `SystemFunction.pyc`, which is a complete compiled version of that module.

Making System Functions

The module `SystemFunction` contains a function that constructs a new instance of the `SystemFunction` class, given a characterization of that system function as a difference equation. Assume that the difference equation has the form

$$a_0y[n] + a_1y[n-1] + ... + a_ky[n-k] = b_0x[n] + b_1x[n-1] + ... + b_jx[n-j] \ ,$$

where X is the input signal and Y is the output signal. It is crucial that the first coefficient of both polynomials be from the same time step; but they can go back different depths in history.

To make a new system function, say corresponding to the difference equation

$$9y[n] - 4y[n-1] + 5y[n-3] = 2x[n-1] \ ,$$

we'd make the call

```
> sf = systemFunctionFromDifferenceEquation([9, -4, 5], [0, 2])
```

You can see from printing out the system function, that we store it as a ratio of polynomials in R:

```
> sf
SF(2.000R/5.000 R**2 + -4.000R + 9.000)
```

Question 1: Describe in English what this system does:

```
systemFunctionFromDifferenceEquation([1], [0, 0, 1])
```

Question 2: What call would you make to generate the system function for a system with this difference equation:

$$y[n] = 2y[n-2] + 3x[n-3] \ ?$$

Understanding a system

A crucial thing to understand about a system is whether, even in the absence of input, its output will go to zero, or whether it will grow unboundedly. We can find this out by determining the *poles* of the system. So, for example, if you ask for the poles of a system function, you'll get a report on the poles of the system and their implications for the stability of the system:

```
> sf = systemFunctionFromDifferenceEquation([9, -4, 5], [0, 2])
> sf.poles()
Magnitude of poles:  [0.7453559924999299, 0.7453559924999299]
[(0.22222222222222227+0.7114582486036491j),
 (0.22222222222222218-0.7114582486036491j)]

> sf2 = systemFunctionFromDifferenceEquation([3, -4, 5], [0, 2])
> sf.poles()
Magnitude of poles:  [1.2909944487358056, 1.2909944487358056]
Danger.  Warning.  Unstable system.
[(0.6666666666666663+1.1055415967851332j),
 (0.6666666666666663-1.1055415967851332j)]
```

Another way to get an understanding of how a system is working is to start it at some initial conditions, and simulate it by running the associated difference equation forward. We have built facilities for doing this into the `SystemFunction` class.

Let k be the order of the polynomial on the Y terms (also called the order of the system) and j be the order of the polynomial on the X terms. We need j initial input values and k initial output values in order to be able to specify all future values of the system. In addition, we need to provide an input sequence; for generality, instead of providing a list, we use a function from n to x[n].

So, given initial values (the lists $[x[0], \ldots, x[j-1]]$ and $[y[0], \ldots, y[k-1]]$) and an input source, the `SystemFunction.plotSequence` method plots the sequence of Y values.

So, for example, you can do

```
> sf2 = systemFunctionFromDifferenceEquation([3, -4, 5], [0, 2])
> sf2.plotSequence([1], [0, 1], lambda x: 1, n = 30, idle = True)
```

and get the plot shown in figure 1. The last two arguments are optional. `n` controls the number of points that are plotted; `idle` controls whether the plotting will work nicely with Idle. *If you call this with* `idle` *set to* `True`*, then you will have to close the plotting window before your program that called this method (or the Python shell) can continue to run.*

Question 3: Experiment with plots of `sf2` above with different initial conditions and inputs.

Robot meets wall

Now, let's return to the problem of the robot driving up to the wall. We would like to determine whether this system is stable. Doing so will require several steps, laid out in the questions below.

Question 4: Imagine a robot driving straight toward a wall. We would like to use a difference equation to model this system, where the output, D, is the distance from the robot to the wall in meters, and the input, V is the robot's current forward velocity in meters per second. Assume that each state transition corresponds to the passage of 0.2 seconds, and that D at time t depends on V at time $t-1$. Write a difference equation to model this system. (Do this on paper).