ECE 300
Signals and Systems
Homework 3

Due Date: Thursday March 26, 2009 *at the beginning of class*

Reading Roberts pages 54-58, 115-140

Problems

1) Consider the following mathematical models of systems:

a) $y(t) = e^{-(t+1)}x(t)$ **b)** $y(t) = \int_{-\infty}^{t} e^{-(t-\lambda)}x(\lambda)d\lambda$ **c)** $y(t) = \int_{0}^{t} e^{-(t-\lambda)}x(\lambda)d\lambda$

d) $y(t) = x\left(1-\frac{t}{2}\right)$ **e)** $y(t) = \int_{-\infty}^{t/2} x(\lambda)d\lambda$ f) $\frac{dy(t)}{dt} = 3y(t) + 2x(t)$

g) $\frac{dy(t)}{dt} = -2ty(t) + 2x(t)$ **h)** $y(t) = \int_{-\infty}^{t} (t-\lambda)x(\lambda+1)d\lambda$

Fill in the following table (Y or N for each question) for each system. You must justify your answers to receive credit. *Assume* t *can be any possible value.*

Part	Causal?	Memoryless?	Linear?	Time Invariant?
a				
b				
c				
d				
e				
f				
g				
h				

For part **f** you should show $y(t) = y(t_0)e^{3(t-t_0)} + \int_{t_0}^{t} 2e^{3(t-\lambda)}x(\lambda)d\lambda$ in order to determine the system is or is not causal and has memory or is memoryless.

For part **g** you should solve the DE first (see handout about integrating factors) and then determine whether the system is or is not causal and has memory or is memoryless.

Remember that we always assume the initial conditions are zero and the initial time is $-\infty$ when determining time-invariance and linearity.

2) (**Matlab Problem**) The ***average value*** of a function x(t) is defined as

$$\overline{x} = \frac{1}{b-a}\int_a^b x(t)dt$$

and the ***root-mean-square (rms)*** value of a function is defined as

$$x_{rms} = \sqrt{\frac{1}{b-a}\int_a^b x^2(t)dt}$$

Read the **Appendix (**located at the end of this homework), then

a) use Matlab to find the average and rms values of the function $x(t) = t^2$ for $-1 < t < 1$

b) use Matlab to find the average and rms values of the following functions

$$\begin{aligned} x(t) &= \cos(t) \quad 0 < t < \pi \\ x(t) &= \cos(t) \quad 0 < t < 2\pi \\ x(t) &= |t| \quad -1 < t < 1 \\ x(t) &= t\cos(t) \quad -2 < t < 4 \end{aligned}$$

Hint: You will probably find the **sqrt** function useful*. You should write a Matlab m-file for this problem, and turn it in with your homework, as well as the answers.*

3) Matlab/Pre-Lab: Experimentation with Sound, Small Signal Models

The MATLAB program **linear_systems_and_sounds.m** allows you to specify an input signal (lines 11-12) and system (line 24), and listen to how the sound changes as the signal passes through the system. We will use the sounds produced to try to identify the systems as linear or nonlinear. Recall from class that a system is linear if and only if an input signal at frequency f produces an output signal at frequency f. The phase angle and amplitude between the input and output may change, but the frequency cannot. Just think about how transfer functions affected periodic signals when you were using phasors to represent sinusoids- the transfer function could change the magnitude and phase but not the frequency of the input signal. *Note that we can only hear sounds when the speaker is vibrating.*

a) Using the code as is (i.e. with y=x), investigate how the frequency of the sinusoidal input signal affects the sound of the signal. Specifically, describe the

sound you hear for input signal frequencies f_0 of 200, 1000, and 5000 Hz.. Note that you need to run the program, wait for the input sound (and graph) to appear, then hit enter and you will hear the output sound and see the output plotted in the lower panel.

b) Now change the system so the output is zero, i.e., y = 0*x (this will make sure the y vector is the correct length). What do you hear for the system output, if anything?

c) Now change the system so the output is y=0*x+10; What do you hear? How fast (at what frequency) is the speaker vibrating?

d) For each of the mathematical models of systems in the following table, assume x is a periodic signal with frequency 200 Hz. For each of these models, determine if the mathematical model of the system is linear or not, and if you would classify the system models as linear or not based on both listening to and looking at the input and output. Discrete-time solutions for model (j) is included in the code, and is commented out. Uncomment the correct lines when you need to run these models.

e) For many system, such as transistors, we are really interested in ***small signal models***. For these systems we only care about the behavior of the system about a set or bias point for an input with a small amplitude. For $x(t)$ small, we can use a Taylor series approximation of $y(x)$ around x(t) = 0 of the form

$$y(x) \approx y_0 + \nu x(t)$$

where y_0 and ν are determined by the Taylor series expansion of $y(x)$ about the point $x = 0$. If we subtract the bias point (y_0) from both sides of the equation and rename variables we have

$$y(x) - y_0 \equiv \Delta y(x) = \nu x(t)$$

This is now a description of a linear system between input $x(t)$ and output $\Delta y(x)$. This relationship is only approximately linear if the input $x(t)$ is sufficiently small that the Taylor series approximation is valid.

For the systems that are mathematically nonlinear, but appeared/sounded linear, determine the small signal linear approximation, that is, determine y_0 and ν.

Note that our input amplitude is 1, and hence something like $z = x/10$ would be considered small. The following formula's for Taylor series about the point $z = 0$ may be helpful:

Taylor series (general form) $f(z) \approx f(0) + z \left.\frac{df}{dz}\right|_{z=0}$

$$y(z) = \left[a + bz\right]^c \approx a^c + ca^{c-1}bz \qquad y(z) = \sin(az) \approx az$$

$$y(z) = \log(a + bz) \;=\; \log(a) \;+\; \frac{bz}{a+b} \qquad y(z) = e^{az} \approx 1 + az$$

(Note that log here means natural log, typing log in Matlab takes the natural log of the function.)

As a check, for those systems where the small signal model is valid for this particular input signal , if you look at the plots from *linear_systems_and _sounds.m*, the output signal should be centered at y_o and should oscillate between $y_o + v$ and $y_o - v$ (or reasonably close to this) with the same frequency as the input signal.

Don't forget that if you want to process element by element, you may need to use ., ./, or .^ in Matlab*

Fill-in and turn in the Table on the following page. You do not need to formally determine if the system is linear or not, you should be able to just look at it (however you can use a formal method if you wish).

Turn in your plots for all of the instances where the small signal model is valid (the system appears to be linear). Show your work in deriving the small signal model, do not just look at the systemOn these graphs indicate that your small signal model has the correct bias (DC point) and the correct swing ($\pm v$).

Remember, if the system is linear and the input is a sinusoid, the output must also be a sinusoid, not just periodic!

System	Mathematical Model	Mathematical Model Linear? (Y/N)	Sounds/Looks Linear?(Y/N)	Small Signal Model (if valid)
a	$y(t)=\frac{1}{2+x(t)}$			
b	$y(t)=\frac{1}{2+\frac{x(t)}{10}}$			
c	$y(t)=\frac{1}{1+\frac{x(t)^2}{10}}$			
d	$y(t)=\sin(2x(t))$			
e	$y(t)=\sin\left(\frac{x(t)}{10}\right)$			
f	$y(t)=\lvert x(t)\rvert$			
g	$y(t)=e^{\frac{x(t)}{10}}$			
h	$y(t)=\log(1.1+x(t))$			
i	$y(t)=\left(10.1+\frac{x(t)}{10}\right)^{0.2}$			
j	$\dot{y}(t)+ay(t)=bx^2(t)$			

Appendix

Maple is often used for symbolically integrating a function. Sometimes, though, what we really care about is the numerical value of the integral. Rather than integrating symbolically, we might want to just use numerical integration to evaluate the integral. Since we are going to be using Matlab a great deal in this course, in this appendix we will learn to use one of Matlab's built-in functions for numerical integration. In order to efficiently use this function, we will learn how to construct what are called *anonymous* functions. We will then use this information to determine the average and rms value of a function. Some of this is going to seem a bit strange at first, so just try and learn from the examples.

Numerical Integration in Matlab Let's assume we want to numerically integrate the following:

$$I = \int_0^{2\pi} (t^2 + 2)dt$$

In order to do numerical integration in Matlab, we will use the built-in command **quadl.** The ***arguments*** to quadl, e.g., the information passed to quadl, are

- A function which represents the integrand (the function which is being integrated). Let's call the integrand $x(t)$. This function must be written in such a way that it returns the value of $x(t)$ at each time t. Clearly here $x(t) = t^2 + 2$
- The lower limit of integration, here that would be 0
- The upper limit of integration, here that would be 2π

Note that an optional fourth argument is the tolerance, which defaults to 10^{-6}. When the function value is very small, or the integration time is very small, you will have to change this.

Anonymous Functions Let's assume we wanted to use Matlab to construct the function $x(t) = t^2 + 2$. We can do this by creating what Matlab calls an **anonymous function**. To do this, we type into Matlab

```
x = @(t) t.*t+2;
```

If we want the value of $x(t)$ at $t = 2$, we just type x(2)

Hence, to evaluate the integral $I = \int_0^{2\pi} (t^2 + 2)dt$ in Matlab we would type

```
x = @(t) t.*t+2;
I = quadl(x,0,2*pi)
```

Note that it is important to define x ***before*** it is used by (passed to) quadl

Example 1 To numerically evaluate $I = \int_{-1}^{1} e^{-t}\cos(2t)dt$ we could type

```
x = @(t) exp(-t).*cos(2*t);
I = quadl(x,-1,1);
```

Example 2 To numerically evaluate $I = \int_{-2}^{1} |t|\, e^{-|t|}dt$ we could type

```
y = @(t) abs(t).*exp(-abs(t));
I = quadl(y,-2,1);
```

Integrating Products of Functions Sometimes we are going to want to integrate the product of functions. While we could just multiply the functions together, it is usually easier to let Matlab do it for us.

Let's assume we want to evaluate the integral $I = \int_{0}^{1} x(t)y(t)dt$, and let's assume that we already have anonymous functions x and y. The function **quadl** needs to be passed a function which is the product of x and y. To do this, we make a new anonymous function z, using the following:

```
z = @(t) x(t).*y(t);
```

and then perform the integration

```
I = quadl(z,0,1)
```

An alternative is to write

```
I = quadl(@(t) x(t).*y(t),0,1);
```

Example 3 To numerically evaluate $I = \int_{-1}^{1} e^{-t}\cos(2t)dt$ we could type

```
x = @(t) exp(-t)
y = @(t) cos(2*t);
z = @(t) x(t).*y(t);
I = quadl(z,-1,1);
```

or

```
I = quadl(@(t) x(t).*y(t),-1,1);
```

Example 4 To numerically evaluate $I = \int_{-2}^{1} |t| e^{-|t|} dt$ we could type

```
x = @(t) abs(t);
y = @(t) exp(-abs(t));
z = @(t) x(t).*y(t);
I = quadl(z,-2,1);
```

or

```
I = quadl(@(t) x(t).*y(t),-2,1);
```