ECE 300
Signals and Systems
Homework 2

Due Date: Tuesday September 18, 2007 *at the beginning of class*

Reading: ZTF pages 28-56 and your course notes.

Problems

1) ZTF, Problem 1-41.

2) ZTF, Problem 1-42.

3) Consider the following mathematical models of systems:

a $y(t) = x(t) + 1$ **b** $y(t) =| x(t) |$ **c** $\frac{dy(t)}{dt} = y(t)x(t)$ **d** $\frac{dy(t)}{dt} = 3y(t) + 2x(t)$

e $y(t) = \sin\left(x(t)\right)$ **f** $y(t) = u(t)$ **g** $y(t) = e^{-t}x(t)$ **h** $y(t) = \begin{cases} x(t) & | x(t) | \leq 10 \\ 10 & | x(t) | > 10 \end{cases}$

i $y(t) = \int_{-\infty}^{t} (t-\lambda)x(\lambda)d\lambda$ **j** $\frac{dy(t)}{dt} = 3ty(t) + 2x(t)$ **k** $y(t) = x(t - t_0) \quad t_0 > 0$

l $y(t) = x\left(\frac{t}{3}\right) + 2$ **m** $y(t) = e^{t}\int_{-\infty}^{t} e^{-\lambda}x(\lambda - c)d\lambda$**,** $c > 0$

Fill in the following table (Y or N for each question) for each system. You must justify your answers to receive credit.

For **e**, look at the case when $x(t)$ is large, and then when $x(t)$ is assumed to be sufficiently small that you can use a small angle approximation.

Part	Causal?	Memoryless?	Linear?	Time Invariant?
a				
c				
d				
e- $x(t)$ large				
e- $x(t)$ small				
g				
j				
k				
l				
m				

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

For part **d** 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 **j** 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. Your solution to the DE will have a similar form to that given for part (d).

4) (**<u>Matlab Problem</u>**) The ***<u>average value</u>*** of a function x(t) is defined as

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

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

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

Read the **Appendix**, 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

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

Hint: You will probably find the sqrt function useful.

For this problem you can just copy down the answers from the Matlab screen. We will assume you are doing this in Matlab because if you are not it will soon become obvious….

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);
```