

Problem Set #4

Due May 3

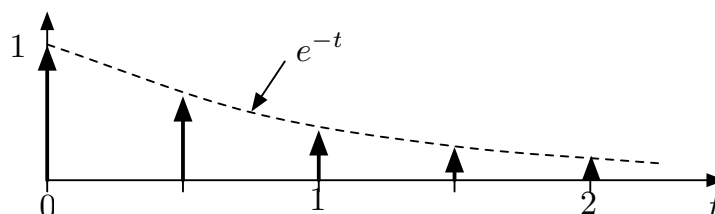
This week the lab will be a little longer than usual, so we'll make the problems shorter. Problems 1 and 2 you can do right away, problems 3 and 4 might be clearer after class on Monday. We are looking at having the midterm in class during the week of May 6th-10th. Let us know if any of these dates are better or worse for you. We're pretty flexible, and would like to find a time that works well for you.

1. Find the Laplace transform of the following signals:

(a) $f(t) = e^{-2t} (\cos(t) + \sin(t))$

(b) $f(t) = (1 - t^2)e^{-2t}$.

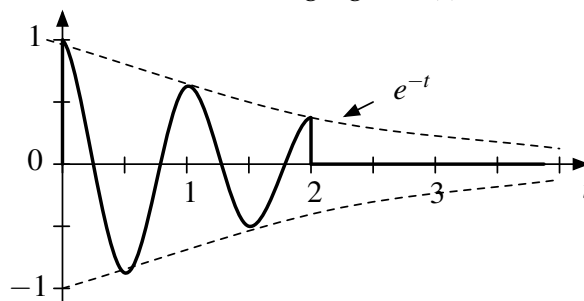
(c) Find the Laplace transform of the following signal $y(t)$



This is an infinite causal sequence of impulses multiplied by e^{-t} .

Recall that $\sum_{n=0}^{\infty} a^n = \frac{1}{1-a}$ when simplifying your answer. Note that this is a *Laplace* transform, so frequency domain convolution is not useful here.

(d) Find the Laplace transform of the following signal $f(t)$



This is two cycles of a cosine, weighted by a decaying exponential. There should be no convolutions in your answer.

2. Find the inverse Laplace transforms of the following functions

(a) $\frac{s+3}{(s+1)^2(s+2)}$

(b) $\frac{10}{(s+1)(s^2+4s+13)}$

(c) $\frac{s^2+8s}{s^2+8s+25}$

3. *Solving Differential Equations*

A system is described by the differential equation

$$y''(t) + 4y'(t) + 13y(t) = 13x(t)$$

where the initial conditions are all zero, $y''(0) = 0$, $y'(0) = 0$, and $y(0) = 0$. The input is the unit step

$$x(t) = u(t).$$

Find $y(t)$.

4. For each of these assertions, determine whether they are true or false. Provide an argument for your conclusion. Remember that an unstable system has poles either in the right-half plane (diverging solutions) or on the $j\omega$ axis (oscillating or constant solutions).

a) Let $h(t)$ be the impulse response of a stable, causal system. Then $\frac{d}{dt}h(t)$ is also stable.

b) Let $h(t)$ be the impulse response of a stable, causal system. Then $\int_{-\infty}^t h(\tau)d\tau$ must be unstable.

c) $H(s)$ is the transfer function of a stable, causal system. The zeros of $H(s)$ must be in the right-half plane for the inverse system $H_{inv}(s)$ to be stable.

Laboratory

This week we will look at the encoding and decoding of touch-tone dialing signals.

The touch-tone system uses signals of different frequencies to encode which button has been pressed. Two frequencies are played at the same time. One frequency encodes the row of the key that has been pressed. The other corresponds to the column. This is known as DTMF, which stands for Dual Tone Multi-Frequency. The frequencies are given in Table ?? . For example, when the button 6 is pressed, the result is the signal

$$y[nT] = \cos(2\pi 770nT) + \cos(2\pi 1477nT).$$

to indicate that key 6 is in the second row, and third column.

Task 1: *Generate a DTMF signal for an input phone number.*

Given an input phone number in the form

```
>> phone_number = [ 1 6 5 0 5 5 5 1 2 3 4];
```

generate the touch-tone signal. Assume each tone is played for 0.5 s, that there is a gap of 0.125 s between tones, and that the sampling rate is 8192 Hz. Implement this as an m-file with the definition

```
>> function ttsignal = ttdial(phone_number)
```

Enter your own phone number and play the resulting signal with `sound()`. Compare it to the sound when you dial your number on the phone.

Task 2: *Use the FFT to estimate the frequencies in a single button push*

We can use the DFT to compute the coefficients of the DTFS of a signal. The N-point DFT computes the DTFS at frequencies 0 to $f_s(N-1)/N$ in steps of f_s/N . Generate signals for the single digits 0, 3, 5, and 7, and plot the spectrum with

```
>> plot(f,abs(fft(d5)/length(d5)))
>> axis([500 1500 0 1]);
```

		f_c		
		1209	1336	1477
$f_r,$	697	1	2	3
	770	4	5	6
	852	7	8	9
	941	*	0	#

Table 1: Frequencies for DTMF encoding of the telephone keypad. If the user pushes key 6, a tone is generated with sinusoid at 770 Hz, to indicate row 2, plus a sinusoid at 1477 Hz, to indicate column 3.

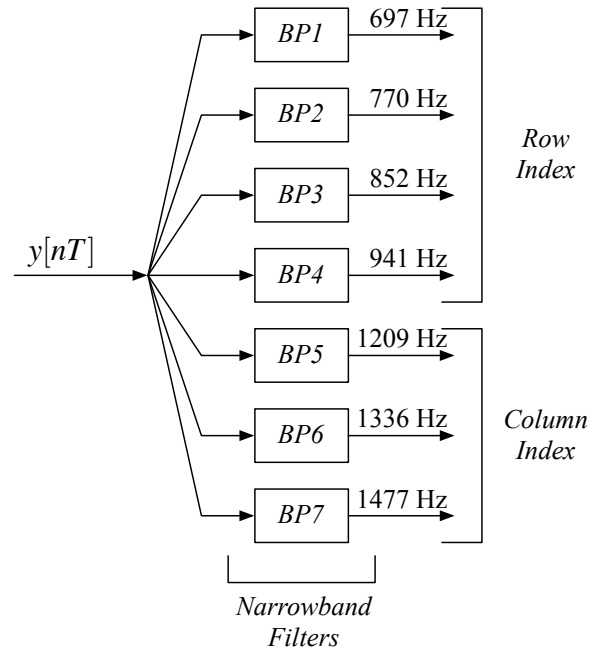


Figure 1: Bank of narrowband filters for decoding the DTMF signal

where \mathbf{f} is the scaled frequency vector, and \mathbf{d}_5 is the signal produced when the key 5 is pressed. The factor $\text{length}(\mathbf{d}_5)$ is due to the fact that, as was pointed out in class, the DTFS has a factor of $1/N$ in the analysis equation, and the DFT has the $1/N$ factor in the synthesis equation. For our application, we're only interested in what frequencies are present, so this scaling doesn't really matter. The axis command zooms in on the part of the spectrum of interest, from 500 Hz to 1500 Hz.

Verify that the correct frequencies have been generated, and plot the spectrum for each of the four buttons.

Task 3: Use a bank of narrowband filters to detect the seven frequencies

The approach used in Task 2 allows us to identify any frequency in the input waveform. In practice, we know that the signal only contains seven frequencies, four for the rows, and three for the columns. A simpler way to process the data is to pass it through a bank of narrowband filters for each of the seven frequencies. By logically combining the outputs of these filters we can determine keys that have been pressed.

1. The narrow band filters should pass one of the frequencies, and suppress the others. The spacing between the frequencies is as small as 73 Hz. We will use a window function as a lowpass filter, and then modulate it to produce a narrowband bandpass filter at the frequencies of interest.

First verify that a 256 sample Hamming window is sufficiently selective. As a first attempt, we generate the window, and compute its spectrum,

```
>> h = hamming(256)';
>> f = [0:255]*8192/256;
>> plot(f, abs(fft(h))/256);
```

Then set the axis to zoom on the frequency interval from 0 to 200 Hz. What is wrong with this plot?

To get a better plot, we need to improve the frequency resolution. Since the resolution is f_s/N , and f_s is fixed, we need to increase N . We do this by adding zeros

```
>> hp = zeros(1,1024);
>> hp(1:256) = h;
>> fp = [0:1023]*8192/1024;
>> plot(fp, abs(fft(hp))/1024);
```

This doesn't add any new information, it just interpolates the previous plot. However, it makes the response much more recognizable. Again, set the axis to zoom in on the frequency interval from 0 to 200 Hz. How much will this filter suppress a signal 73 Hz away? Give your answer as a fraction, and in dB.

2. Next we will generate the narrowband bandpass filters by modulating the Hamming window by a cosine of the desired frequency

```
>> tf = [0:255]/8192;
>> h1 = h.*cos(2*pi*f1*tf);
```

where $h1$ is the filter for the first frequency, $f1$. Generate a 7 by 256 element array of these filters, with each row corresponding to one of the seven frequencies.

Take the signal from your phone number in Task 1, and filter it (use `conv()`) with each of the seven bandpass filters. Plot the response for one of the row frequencies and one of the column frequencies that appear in your number.

3. In the previous case we modulated the Hamming window with a cosine, and the result was modulation in the output. In general we don't know when the button was pushed, and hence, what the phase of the cosine is. When the input and the filter are 90 degrees out of phase, the output is zero. As the filter shifts in the convolution operation, the filter and the input cosine are periodically going in and out of phase. We would like to avoid needing to know what the phase of the input is. We really only want the envelope.

One alternative is to rectify the filtered signals, and then lowpass filter, as in the AM receiver that was described in 102A (and 102).

Another approach here is to use complex modulation

```
>> h1e = h.*exp(i*2*pi*f1*tf);
```

This effectively only filters out one of the two complex exponentials that make up a sinusoid. The result has a constant magnitude response for any relative phase between the filter and the input.

Generate a 7 by 256 element array of these filters, with each row corresponding to one of the seven frequencies. Filter the signal you created in Task 1. Plot the magnitude of the result for the same row and column frequency that you used before.

Task 4: *Decode the touch-tone waveform*

You now have all the information you need to decode the touch-tone waveform. Assume that the digits and the quiet period are at least 0.125 s, but could be longer. Write an m-file that takes the touch-tone signal as an input and

1. Finds the quiet separators,
2. Finds the intervals with key presses,
3. Identifies the two largest frequencies in each interval,
4. Looks up the digits these corresponds to, and
5. Returns the decoded phone number.

Several test cases are provided on the EE102B web site, in the `t_test_cases.mat` file. Try your decoder on these waveforms, and report the results.