

ECE 220 SIGNALS & SYSTEMS I
Laboratory Project 4
Spring 2008

Assigned: February 20/21, 2008

Report Due: March 19/20, 2008

The purpose of this project is to learn how to use Matlab to implement the convolution integral numerically. You will gain more practice with convolution, and see an application of convolution in the implementation of matched filters.

Your report for this project will consist of all the analytical (*i.e.*, pencil/paper) work, Matlab plots and code, and relevant explanations. A list of guidelines for preparing the lab report is posted on the ECE 220 website. Each student must do his or her own work on this project, however you may ask other students or any of the teaching staff for advice. As stated in the guidelines given in the ECE 220 course information packet, you should identify any students you talk to about the project.

1 Prelab

Before starting on the lab exercises below, review section 2.2.1 in the Signals and Systems text by Oppenheim/Willsky/Nawab. Also prior to beginning the lab, determine and sketch the convolution $y(t) = x(t) * h(t)$ when $h(t)$ and $x(t)$ are as defined below:

$$h(t) = u(t) \qquad x(t) = e^{-t}u(t).$$

2 Numerical Approximation to Convolution

The continuous-time convolution integral is defined as follows:

$$y(t) = \int_{-\infty}^{+\infty} x(\tau)h(t - \tau)d\tau. \tag{1}$$

In order to use Matlab to implement the convolution integral, it is helpful to model the signals $x(t)$ and $h(t)$ as piecewise-constant functions. Following section 2.2.1 of the Oppenheim/Willsky/Nawab textbook, the signal $x(t)$ can be approximated by $\hat{x}(t)$:

$$\hat{x}(t) = \sum_{k=-\infty}^{+\infty} x(k\Delta)\delta_{\Delta}(t - k\Delta), \tag{2}$$

where the short rectangular pulse $\delta_{\Delta}(t)$ is defined as follows

$$\delta_{\Delta}(t) = \begin{cases} 1 & -\Delta/2 \leq t < \Delta/2 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Note that the definition of $\delta_{\Delta}(t)$ in Equation 3 differs slightly from that of the textbook, but the basic idea is the same. The signal $h(t)$ can be approximated in a similar manner, *i.e.*,

$$\hat{h}(t) = \sum_{k=-\infty}^{+\infty} h(k\Delta)\delta_{\Delta}(t - k\Delta), \tag{4}$$

In other words, $x(t)$ and $h(t)$ are approximated as a series of narrow rectangular pulses with heights determined by the values of the functions sampled at the center of each rectangular pulse. Figure 1 shows $\hat{x}(t)$

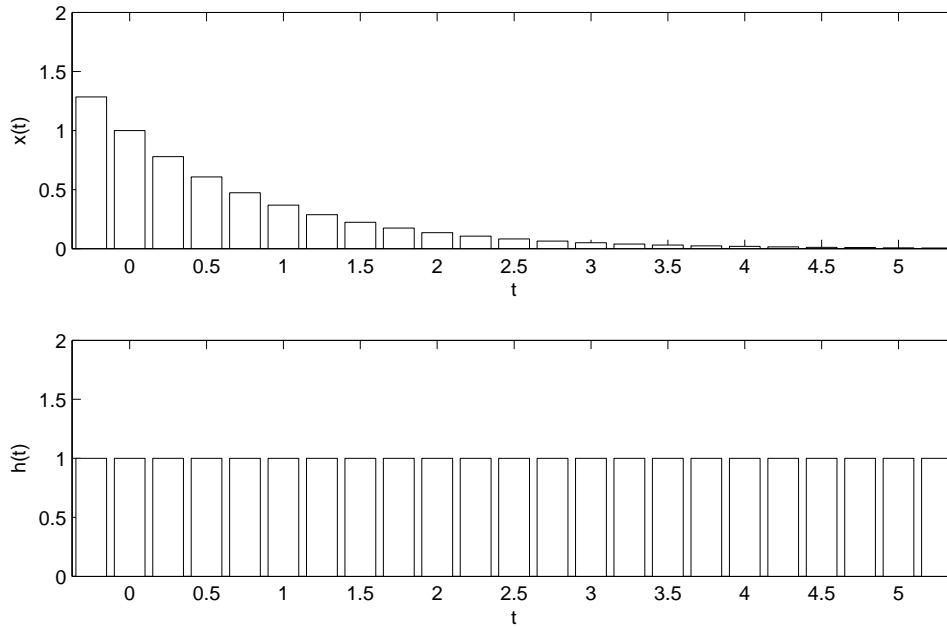


Figure 1: Piecewise-constant approximations to the signals $x(t)$ and $h(t)$. These approximations were computed with $\Delta = 0.25$.

and $\hat{h}(t)$ computed using $\Delta = 0.25$, thus there is a short pulse located every 0.25 seconds. Obviously the approximations will improve as Δ is reduced.

The goal of this section is to compute the convolution of these piecewise-constant signals, *i.e.*, to compute $\hat{y}(t)$ defined as

$$\hat{y}(t) = \int_{-\infty}^{+\infty} \hat{x}(\tau) \hat{h}(t - \tau) d\tau. \quad (5)$$

Matlab can't compute a continuous function, but it can compute samples of a function. Fortunately, if we choose to compute the samples of $\hat{y}(t)$ located Δ seconds apart, the calculations will be simplified substantially. In other words, we'll focus on computing the samples $\hat{y}(n\Delta)$ for $n = 0, 1, 2, \dots$. Consider the equation for $\hat{y}(1\Delta)$:

$$\hat{y}(1\Delta) = \int_{-\infty}^{+\infty} \hat{x}(\tau) \hat{h}(1 - \tau) d\tau. \quad (6)$$

To compute this integral, we need $\hat{h}(1 - \tau)$, which is shown in Figure 2. From Figures 1 and 2 we can see that the piecewise-constant sections of $\hat{x}(\tau)$ and $\hat{h}(1 - \tau)$ are perfectly aligned. Multiplication of $\hat{x}(\tau)$ and $\hat{h}(1 - \tau)$ yields another piecewise-constant function, and computing the integral is very easy to do. (Remember that an integral corresponds to the area under a function, and it is easy to compute areas of rectangles.) Whenever t is equal to a multiple of Δ (*i.e.*, $t = n\Delta$) the functions will be aligned and the integral will be easy to compute.

- (a) Starting from the convolution sum, show how that the desired samples of $\hat{y}(t)$ can be computed as follows

$$\hat{y}(n\Delta) = \Delta \sum_{k=-\infty}^{+\infty} x(k\Delta) h((n - k)\Delta). \quad (7)$$

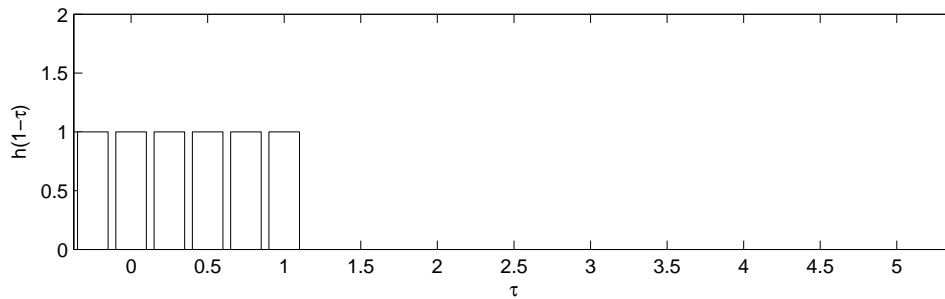


Figure 2: Piecewise-constant approximation to the signal $h(1 - \tau)$. The approximation was computed assuming $\Delta = 0.25$.

Note that this equation should be familiar from your work in ECE 201. It is the discrete convolution sum! Matlab provides a function called `conv` that computes discrete convolution sums. Type `help conv` to read more about this function.

- (b) Define the vector `x1` to be samples of the signal $x(t)$ defined in the prelab for $0 \leq t \leq 10$. Similarly define the vector `h1` to be samples of the signal $h(t)$ for $0 \leq t \leq 10$. (Remember that Matlab indexes its vectors starting at 1, so the $n = 0$ sample will be `x1(1)`). Use the `conv` function to compute `y1` defined by the convolution given in Equation 7.

Note that because `x1` and `h1` are finite-length segments of infinite length signals, only a finite-length segment of `y1` will be valid. For what values of t does `y1` represent a good approximation to the $y(t)$ you found in the prelab?

- (c) Re-do the calculations in part b for several values of Δ : $\Delta = 0.25, 0.1, \text{ and } 0.01$. Plot your results on a single graph, along with the exact results you found in the prelab. Comment on how well the approximations work for different values of Δ .

3 Application: Matched Filtering

In this section you will use the numerical convolution algorithms you derived in the previous section to explore the application of matched filtering.

3.1 Background

The matched filter for a signal $b(t)$ is defined as follows:

$$h_{\text{MF}} = b(-t).$$

When the input consists only of the signal $b(t)$, the output of the matched filter is the autocorrelation $\phi(t)$ of $b(t)$.

- (a) Using the definition of the matched filter and the convolution integral, show that the autocorrelation is given below:

$$\phi(t) = \int_{-\infty}^{+\infty} b(t + \tau)b(\tau)d\tau.$$

It can be shown that the maximum value of the autocorrelation occurs at $t = 0$.¹ For instance, consider the autocorrelation of the a simple square pulse defined by $b_{\text{square}}(t) = u(t) - u(t - 2)$.

- (b) Using convolution, show that the autocorrelation of the square pulse is equal to a triangular pulse centered at the origin.

Matched filters are quite useful in radar systems. In radar an electromagnetic pulse is transmitted. This pulse is reflected by an object of interest (*e.g.*, an airplane) and returns to the transmitter. The signal received at the transmitter is a time-shifted and possibly amplitude-scaled version of the original signal. The time delay is proportional to the distance between the transmitter and the object. A matched filter can be used to measure the time delay, thus it provides an estimate of the distance to the object.

Consider the following scenario. A radar system sends out the signal $b(t)$. This signal bounces off an object, and the returns to the transmitter. The received signal at the transmitter is $r(t)$ given below

$$r(t) = b(t - D),$$

where D is the delay time. In this case we've assumed that the signal is not scaled in amplitude; it is only delayed. Suppose $r(t)$ is processed using a matched filter for the signal $b(t)$. The result is the signal $p(t) = r(t) * h_{\text{MF}}(t)$.

- (c) Determine a general formula for $p(t)$. You should be able to write your answer in terms of $\phi(t)$. *Hint: use the property of time invariance!*
- (d) How could you determine the value of D given a plot of the processed signal $p(t)$? Use the example of the square pulse $b_{\text{square}}(t)$ to illustrate your results.

In the exercises that follow, you will implement matched filters for a set of signals known as Barker codes. Barker codes have been used in pulse compression radar systems and for synchronization in digital communication systems. The book by Van Trees [1] provides additional information on Barker codes.² Figure 3 shows the Barker codes of length 3, 7, and 13. These signals have desirable autocorrelation properties. To see this, do the following:

- (d) Find the autocorrelation of the the length 3 Barker code, *i.e.*, $\phi_3(t) = b_3(t) * b_3(-t)$. You should be able to do this convolution by hand. Include your sketches and other calculations with your lab report.

Note that the autocorrelation for $b_3(t)$ has a maximum peak of 3, and all the other peaks are less than or equal to 1. For all Barker codes (defined as in Figure 3) the maximum peak of the autocorrelation is equal to the length of the signal, and all the other peaks are less than or equal to 1. Having this type of autocorrelation is desirable for applications where the matched filter is used to estimate the time of arrival of a particular signal.

3.2 Creating Barker code signals in Matlab

- (a) Since Barker codes consist of a series of 1's and -1's, they are easy to generate with digital logic. Similarly, it is easy to use Matlab's logic functions to define a Barker code signal. The following code can be used to generate samples of the length 3 Barker code signal $b_3(t)$:

¹This proof is beyond the scope of ECE 220. It is a consequence of the Schwarz inequality. More information may be found in linear algebra textbooks, and in many communications textbooks, such as the one used in ECE 460.

²See chapter 10 of the Van Trees textbook. Note that Dr. Van Trees is a Professor Emeritus in the ECE Department at GMU.

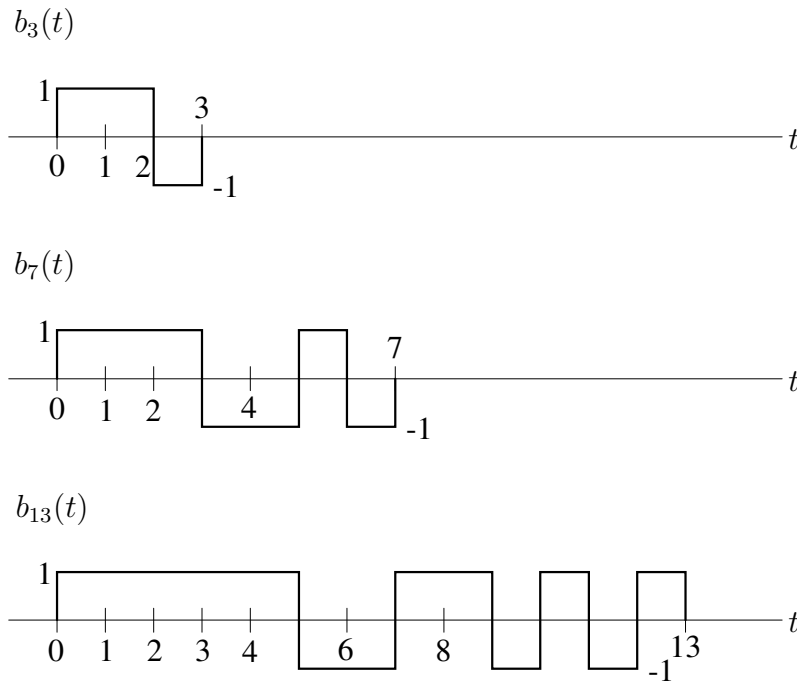


Figure 3: Barker codes of length 3, 7, and 13.

```
t=0:0.01:3;
b3=1*(t>=0 & t<=2)-1*(t>=2 & t<=3);
```

Type this code into Matlab and verify that it generates samples of the signal $b_3(t)$. Plot the signal you generated versus time t and include the plot in your report.

- (b) Using the code described above as a model, write the Matlab code to generate samples of the length 7 and length 13 Barker codes. Use the same sample period (0.01) that you used to generate the length 3 codes. Include plots of the signals $b_7(t)$ and $b_{13}(t)$ in your report.

3.3 Implementing matched filters for Barker codes

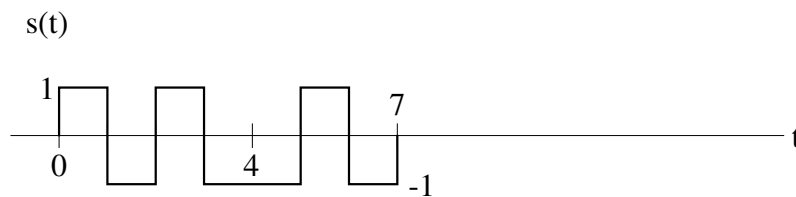
The matched filter for the signal $b(t)$ is defined as $h_{MF}(t) = b(-t)$. Assuming that the signal $b(t)$ is non-zero for $t \geq 0$ it is easy to see that the matched filter will not be a causal system. Since the signal $b(t)$ is finite length, it is possible to create a causal filter by time-shifting the matched filter. This new filter is defined as $h_{MFC} = h_{MF}(t - T)$.

- (a) For each of the 3 Barker codes ($b_3(t)$, $b_7(t)$, and $b_{13}(t)$), determine the value for T required to make their matched filters causal. Suppose that we apply causal matched filter to a signal. How is the result related to the autocorrelation? In other words, if we compute $\phi_{causal}(t) = b(t) * h_{MFC}(t)$, how is ϕ_{causal} related to $\phi(t)$?
- (b) In Matlab it is easy to generate the causal matched filter for a given signal. The `fliplr` command flips a row vector, thereby creating the samples of the signal needed for the matched filter.³ Use the

³If we had defined the signals as column vectors instead of row vectors, we could use the command `flipud` to generate the matched filter.

`fliplr` command to generate the causal matched filters for the 3 Barker code signals. Plot the results and include the plots in your report.

- (c) Compute the causal autocorrelation ($\phi_{\text{causal}}(t)$) for each of the 3 Barker code signals by convolving them with their causal matched filters. Verify that $\phi_{\text{causal}}(t)$ has a peak or the right height at the appropriate time and that the other absolute value of the peaks in the signal are less than or equal to 1 (as we would expect from a Barker code). For the $b_3(t)$ signal, your Matlab results should confirm the results you obtained analytically in Section 3.1 above.
- (d) Not all signals have as nice autocorrelation properties as the Barker codes. Consider the signal $s(t)$ shown below. Define this signal in Matlab and compute its autocorrelation via convolution using the matched filter. What do you observe? Include the plots in your report.



Note that $s(t)$ differs from the length 7 Barker code in just one time interval, yet you should find that its autocorrelation looks significantly different. Specifically, you should see that the absolute value of the peaks around the main peak are larger than for the Barker code signal.

- (e) In realistic radar systems, the received signal $r(t)$ includes a noise component in addition to the signal component, *i.e.*,

$$r(t) = b(t - D) + \text{noise}.$$

Since the matched filter is a linear system, the output due to the noisy input will be

$$p(t) = r(t) * h_{\text{MFC}}(t) = b(t - D) * h_{\text{MFC}}(t) + \text{noise} * h_{\text{MFC}}(t).$$

In other words, there will be a noise component added to the output we would expect from the noise-free matched filtering operation. The file `lab4_mysterysig.mat` posted on the course website contains a vector `noisy_r` and an associated time vector `t`. `noisy_r` is a delayed Barker signal of length 13 that has had noise added to it. Plot this signal. Process it with your causal matched filter for the $b_{13}(t)$ signal, and try to determine the delay time D . Note: in estimating the delay time, you need to account for the fact that you are using the *causal* version of the matched filter, rather than the non-causal matched filter.

Prior to processing the mystery signal, you are encouraged to try embedding a signal in noise and testing your matched filter processing. The following code generates a delayed and noisy length 13 Barker signal:

```
noisysig=[zeros(1,100) b13 zeros(1,200)].+.1*randn(1,1601);
```

Note that the Barker signal here is delayed by 1 second (since we have samples every 0.01 seconds and there are 100 zeros at the beginning of the signal). The `randn` command generates Gaussian random noise. If you've taken a statistics class, you are probably familiar with Gaussian noise. The mystery signal was created using a similar command in Matlab. A different delay value was used to create the signal, and you're trying to estimate this delay.

4 Acknowledgment

The first part of this assignment was motivated by a similar project in the book *Computer Explorations in Signals and Systems Using Matlab* by John R. Buck, Michael M. Daniel, and Andrew C. Singer.

References

- [1] H. L. Van Trees, *Detection, Estimation, and Modulation Theory, Part III*. New York, NY: John Wiley and Sons, 1971.