GEORGIA INSTITUTE OF TECHNOLOGY

SCHOOL OF ELECTRICAL ENGINEERING

ECE 6272 FALL 2010

COMPUTER PROJECT #2

Assigned: Tuesday, September 14, 2010

**Due Date for On-Campus Students: Tuesday October 5 @ 9:35 AM**

**Due Date for Video Students: Tuesday, October 12 @ 4:00 PM Eastern Time**

This project is to be done *individually.* Each student must develop his or her own computer code in its entirety. Students are not to discuss the theory or approaches to coding the theory with one another, nor are they to assist in debugging each other’s work. You may ask Dr. Richards questions regarding theory and implementation of the project, including e-mailing them or asking them at the beginning or end of class or during office hours, when others can benefit as well. MATLAB is the preferred language, but others are acceptable; the point is to try the experiments, not to improve your MATLAB skills.

Reports will be graded on completeness in addressing the assignment and quality of results. They will not be graded on programming style or efficiency or on writing quality, except that the programming and the writing should be clear enough to be reasonably understandable. Questions or clarifications about the assignment should be directed to Dr. Richards.[[1]](#footnote-1) Errata, revisions and hints (if any) will be made available via e-mail to the class or during class hours.

# PROBLEM

In this project, you will explore the basic properties of the linear FM (LFM), or “chirp”, pulse compression waveform and its associated matched filter, and compare them against a simple pulse of the same length and energy. Matched filtering will be implemented in both the time and frequency domains. Weighting for sidelobe control will be implemented. Losses due to weighting will be estimated and compared against theoretical predictions. Finally, you will demonstrate the range-Doppler coupling effect and, possibly, a way to correct range measurements for this effect.

# REPORT FORMAT

To aid in the grading, please observe the following format constraints. All plots should be adequately labeled and identified so I can tell which plot goes with which part of the assignment.

* The first page should be a cover page with the class number, your name, and the project title
* The second page of your report should contain the following specific items:
  + the “three-chirp” spectrum plot, carefully labeled on both axes, from ¶3.2, along with your answer to the questions about BT products and your comments on how well these spectra approximate a rectangle;
* The third page (or two) of your report should contain the following specific items required in ¶3.3:
  + a plot of the matched filter outputs for the two pulses, overlaid and carefully labeled. Answer the questions asked about the simple pulse bandwidth, pulse energies, peak values, and signal energy; and
  + answers to the questions about the width of the filter output peak for the simple and LFM pulses, and consistency with their bandwidths. Include a plot of the LFM matched filter output sampled sufficiently densely to clearly define the first null.
* The fourth page (or two) of your report should contain the following specific items required in ¶3.4:
  + a plot showing that you obtained the same matched filter output using frequency domain techniques as you did with time domain techniques;
  + the plot of the matched filter output with and without the time-domain window, in dB and with the time axis labeled in seconds. Include your answer to the question about *LPG*, and
  + your answer to the question about mainlobe width, with and without the window; and
  + your answer to the question about peak sidelobe level, with and without the window.
* The fifth page (or two) of your report should contain the following specific items required in ¶3.5:
  + the plot of the matched filter output with and without the time-domain window, in dB and with the time axis labeled in seconds. Include your answer to the question about *LPG*, and
  + your answer to the question about mainlobe width, with and without the window; and
  + your answer to the question about peak sidelobe level, with and without the window.
* The sixth page (or two) of your report should contain the following specific items required in ¶3.6:
  + the plot of the matched filter output for the two-target data for both simple and LFM pulses (overlaid on a single plot and labeled in range units);
  + your answer to the questions about actual target ranges, measured ranges with the LFM pulse, and rnage resolution for both pulses relative to the target spacing.
* The seventh page (or two) of your report should contain the following specific items required in ¶3.7:
  + the plot of the matched filter output with a Doppler mismatch and an upchirp waveform;
  + your answer to the questions about the peak shift, if any, and its relation to the predicted value; also the effect on the magnitude; and
  + the results for the down chirp, including answers to the questions about peak shift, comparison to the upchirp case, and how to figure out the real range.
* All code listings should be kept together at the end.
* Other plots, diagrams, derivations, and explanations, if any, should be between these two things. For instance, additional supplementary plots, and anything else needed to make your reasoning and conclusions clear.

Your report must be typed, not handwritten, and should have figures included in the report, not in a separate document. On-campus students should submit a hard copy. If you will be absent from class on the day the report is due, turn it in in advance of the due date and time, either at my office, in my mailbox in the ECE mail room. If you deliver it to my mailbox, ask an ECE staff or faculty member to note the time and date and initial it. If you cannot deliver a hard copy in advance, e-mail me your report by the due date and time.

Distance learning students must e-mail an electronic copy to me. PDF format is preferred, but I can also accept Microsoft Word. Do not send TeX or postscript documents. All text, figures, and code should be in a single PDF (or Word) file.

If you anticipate a problem in submitting your work on time, contact me before it is due, not when it is due or afterwards.

# SIMULATION REQUIREMENTS

## Generating a Complex LFM Chirp Waveform

A complex, continuous-time LFM chirp waveform that sweeps from –**/2 Hz to +**/2 Hz over a duration of**seconds is given by



Note that this is just the modulation function; an actual transmitted waveform would include an RF carrier term as well. The instantaneous frequency of a waveform is defined by



where **(*t*) is the phase function, in this case (**/**)*t*2. Inserting the LFM phase function of Eqn. into Eqn. gives



which sweeps linearly from –**/2 Hz to +**/2 Hz as *t* runs from –**/2 to +**/2, as desired. The *time-bandwidth product* or “BTproduct*”* of the chirp is simply the product of its swept bandwidth and pulse length, **. Equation is called an “up chirp” because the instantaneous frequency *in*creases during the pulse. A “down chirp” would be obtained by simply conjugating the phase of Eqn. .

A discrete time equivalent can be defined by sampling *x*(*t*) at some interval *Ts*. Since the swept instantaneous bandwidth is ** Hz, we might expect that the actual two-sided bandwidth is about ** Hz (we will see that this depends on the BT product). We will therefore assume the sampling rate should be at least ** samples/sec. It is often convenient to oversample the chirp to get better graphical results. We allow for this by including an oversampling factor *k* that will range from 1.2 to 10.0. Thus, the sampling interval becomes 1/*kB* samples/sec. The number of samples in the pulse is then **/*Ts = k*, or *k* times the BT product. Sampling Eqn. at this rate, and also shifting it in time by **/2 seconds to make it causal, gives us the discrete-time LFM chirp



This is the expression we want to work with. A listing of a MATLAB function git\_chirp.m that implements Eqn. is included at the end of this document. The file is available on the class T-Square site in the “Project Data and Solutions” section.

MATLAB has a built in function chirp that can also be used to generate the waveform of Eqn. . chirp generates real, rather than complex, chirp functions. However, by using two calls and changing the starting phase by 90º you can generate the real and imaginary parts of the complex chirp, then combine them to produce the final result. It is probably easier to use git\_chirp.m.

## Effect of BT Product on Chirp Spectrum

The magnitude of the chirp waveform’s Fourier transform is usually considered to be well-approximated by a rectangle function with support on (–**/2,+**/2), provided ** is large enough, typically on the order of 100 or more; that is, we assume



Whether this is a good approximation or not depends on the BT product. To show this, carry out the following steps:

1. Generate three chirp pulses, each of duration ** = 100 **s but with swept bandwidths of 100 kHz, 1 MHz, and 10 MHz. Oversample by a factor of *k* = 1.2. What are the corresponding BT products? Note that while the analog pulses would all be the same length (10 **s), the discrete-time pulses are *not* the same length. As the bandwidth increases, so does the sampling rate; and since the pulse length in seconds is constant, the number of samples is proportional to the bandwidth, as shown in Eqn. .

2. Use the FFT to compute and plot the magnitude of the discrete Fourier transform of all three chirps, preferably on the same plot so that they overlay one another. Use a relatively large FFT size so that you get the discrete time Fourier transform (DTFT) evaluated at a dense set of frequency samples; this will ensure you get a good visual representation of the details of the DTFT. Furthermore, it is desirable to rotate the FFT output so that zero Hz appears in the middle of the plot, rather than at one end. In MATLAB, the function fftshift exists for just this purpose. The frequency axis of your plot must be correctly labeled in normalized radian frequency **, *i.e.* from −** to +** in radians/sample, or in normalized cyclical frequency *f* from −0.5 to +0.5 cycles/sample.

When you generate the desired plot, you should find that all three spectra have the same bandwidth in the plot, even though their bandwidths in Hz varied by 100:1. This occurs because the FFT outputs are in normalized frequency units. Each pulse was sampled at a rate proportional to its bandwidth, so the normalization that converts from frequency in Hz to normalized radians ** is different for each pulse, such that they all have the same bandwidth on the normalized frequency scale. One of the useful things about normalized frequency units is that they allow us to compare the spectra of functions sampled at different sampling rates.

You should also find that the amplitudes of the spectra differ significantly. This is because the “energy” in each discrete-time pulse differs. Specifically, each pulse has an amplitude of 1 for each sample; but the number of samples, and therefore the total energy (sum of the squared magnitudes of the samples) is proportional to the swept bandwidth **. Parseval’s theorem tells us that the energy of a waveform is the same, whether measured in the time or frequency domain. Since the energy is higher for the longer (higher bandwidth) discrete-time pulses, this means that the area under |*X*(**)|2 must increase with **; but since the *normalized* bandwidth is nominally the same for all three cases, the spectrum amplitude must be proportional to . That is, the width is the same for all three spectra in units of **, so the height must increase with ** so that the area (energy) can increase. This also means that if you normalize each sequence (or its FFT) by  before plotting, they should all come out with about the same nominal amplitude. Go back and apply this normalization to obtain the final plot of the spectra in step (2) above. If this goes well, all three spectra should now be about the same width and height in your plot.

Once you have generated your plot, examine the three spectra and comment on how well the magnitude of the Fourier transform of a chirp can be approximated by a rectangle function.

## Matched Filtering and Resolution in the Time Domain; Comparison Against the Simple Pulse

We now compare an LFM with a BT product of 100 with a simple pulse of the same length.

1. Generate an LFM complex chirp with duration ** = 100 **s and swept bandwidth ** = 1 MHz. Oversample by a factor of at least *k* = 10. Your pulse should have a length of *k* = 100*k* samples. Also generate a simple pulse of the same length; this would be simply a vector of *k* ones, which is easily generated with MATLAB’s ones function. What is the approximate bandwidth, in Hz, of the simple pulse?

2. Working in the time domain, compute the output of the matched filter for each of the two pulses, assuming a single point scatterer. This will simply be the convolution of the waveform with its matched filter, or equivalently, the autocorrelation of the waveform. The computation must be done in the time domain. Overlay plots of the magnitude[[2]](#footnote-2) of the two matched filter output responses onto the same plot. The time axis of the plot must be labeled in seconds, not samples.

3. Compare the two responses. The derivation of the matched filter showed that the peak signal component at the output equaled the signal energy, regardless of the detailed shape of the signal. That is, any two signals with the same energy should produce the same peak magnitude at the output of their respective matched filters. Do these two waveforms have the same energy? Do their respective matched filter outputs have the same peak value? Does it equal the signal energy?

4. The first null of the Fourier transform of a rectangular pulse of length *a* (which is a sinc function) is expected to occur at 1/*a* in the complementary Fourier domain. Thus a time-domain simple pulse of length ** seconds has a sinc Fourier transform with a first zero at 1/** Hz (Rayleigh resolution in frequency), while a rectangular spectrum of width ** Hz has a sinc function with a first zero at 1/** seconds in the time domain as its inverse Fourier transform (Rayleigh resolution in time).[[3]](#footnote-3) We also know that the 3 dB bandwidth of the sinc will be 0.89/*a*; thus the peak-to-null bandwidth is also somewhat indicative of the 3-dB width of the sinc pulse, and it is easier to measure. Continuing with the comparison of the LFM and rectangular pulses, what is the Rayleigh width (in seconds) of the mainlobe response of the simple pulse? Of the LFM chirp? Is this consistent with their respective bandwidths? You may find that you get much better results if you computed the results above with a high oversampling factor. A factor of at least 10 is required. The higher oversampling factor will give you much more detail in the mainlobe and help you avoid being misled about the first zero location due to a sparsely-sampled filter output, but will generate longer signal vectors and take longer to run. However, for most current computers, run time should not be much of an issue.

## LFM Sidelobe Suppression Using Frequency Domain Techniques

You are probably used to the idea of using a window on a filter impulse response in order to improve the stopband attenuation, or on time-domain data to reduce the sidelobes of the DTFT. In either case, to reduce sidelobes in the frequency domain you apply weighting in the time domain. Similarly, to reduce sidelobes at the output of the matched filter, *i.e.* in the time domain, we must apply weighting in the frequency domain.

It is required that you use an oversampling factor of at least *k* = 10, and possibly as much as  
*k* = 30*,* throughout this section to get good definition of the sidelobe peaks.

1. Use the same chirp you utilized in Section 3.3. Again perform matched filtering of the pulse, but this time in the frequency domain, that is, using FFTs. Before proceeding further, verify that you obtain the same matched filter output that you obtained using time domain techniques in Section 3.3.

2. The Hamming window function is



This function is available in the Signal Processing Toolbox of MATLAB as hamming.m. If you do not have this toolbox or are using a different programming environment, simply create your own Hamming window using the formula above. Then incorporate this window function into the matched filtering process by applying it appropriately in the frequency domain. Give some thought as to how much of the spectrum it should be applied to, and how it should be aligned with respect to the waveform spectrum. *It is highly advisable* *to plot the spectrum before windowing, along with the window itself on the same plot, and think about how you want to align the window with respect to the LFM spectrum.* There is not necessarily only one reasonable choice regarding the window extent, but all reasonable choices are probably similar to one another.

3. Plot the output of the matched filter with and without the window included, preferably overlaid on the same plot. Use a decibel scale for the amplitude; the time axis must be labeled in seconds. Do *not* normalize the magnitudes to have the same peak response. You should see a reduction in the amplitude of the mainlobe peak when you use the window function. How much is this reduction, called the *loss in processing gain* (LPG) in dB? How does this compare to the predicted value[[4]](#footnote-4)



4. What is the peak (not necessarily the first) sidelobe level (in dB relative to the corresponding mainlobe peak) with and without the window? What is the approximate peak-to-first null width (Rayleigh resolution) of the mainlobe response in seconds, both with and without the windowing?

## LFM Sidelobe Suppression Using Time Domain Techniques

As shown in class, for LFM waveforms specifically, it is possible to shape the spectrum of the matched filter for sidelobe reduction by windowing the impulse response instead of the frequency response; in other words, by applying the window in the time domain instead of the frequency domain. This doesn’t work for arbitrary waveforms, but it does work for linear FM.

Again use an oversampling factor of at least *k* = 10, and possibly as much as *k* = 30*,* throughout this section to get good definition of the sidelobe peaks.

1. This time incorporate this window function into the matched filtering process by applying it in the time domain to the impulse response of the matched filter.

2. Plot the output of the matched filter with and without the window included, preferably overlaid on the same plot. Use a decibel scale for the amplitude; the time axis must be labeled in seconds. Do *not* normalize the magnitudes to have the same peak response. You should again see a reduction in the amplitude of the mainlobe peak when you use the window function, since the impulse response is no longer exactly matched to the radar waveform. How much is the LPG in dB? How does this compare to the predicted value you found above, and also to the observed value when you used frequency domain windowing?

3. What is the peak sidelobe level (in dB relative to the corresponding mainlobe peak) with and without the window? Compare the peak sidelobe value with the window to the corresponding value when you windowed in the frequency domain. What is the approximate peak-to-first null width (Rayleigh resolution) of the mainlobe response in seconds, both with and without the windowing?

## All-Range Matched Filtering and Two-Target Resolution

Now that you know how to do matched filtering in the time domain with simple and LFM pulses, and how to control LFM time (range) sidelobes, let’s apply that to a situation with multiple targets. For this section, use an LFM pulse with a duration of ** = 100**s and bandwidth ** = 1 MHz, and choose an oversampling factor of *k* = 2. Thus, the fast-time sampling rate is 2 Msamples/sec, so the samples are 0.5 **s apart, corresponding to 75 m spacing of the samples in range. Also create a simple pulse, say *xs*[*n*] of the same length and amplitude as your LFM pulse.

With a 100 **s pulse, the radar’s minimum range is 15 km. Assume the radar is configured to measure a range swath extending from 20 km to 50 km. This is done by simply not starting to collect range samples until 2(20 km)/*c* = 133.33 **s after the leading edge of the pulse is transmitted, and stopping after 333.33 **s. The number of samples in this 200 **s interval is 401.

Now create an artificial echo containing targets at two different ranges. To do this, create a vector of 401 zeros to represent your range bin data. Add a copy of the simple pulse *xs*[*n*] into this vector beginning at sample number 101. (I’m using MATLAB-style indexing here, so the first sample of the vector is #1, not #0.) Add a second copy beginning at sample number 141. The two pulses will overlap, so be sure to add this second copy to the existing data, not overwrite it. The total echo from the two targets is the superposition of the two individual echoes. What are the ranges of the two artificial targets that would result in their echoes showing up in these range bins?

Now matched filter your two-target, simple pulse data vector with the appropriate matched filter. Plot the output waveform, being careful to correctly label the horizontal axis in units of range. To do that, you must think about the range to the initial sample, the spacing between range samples, the delay through the matched filter, and the MATLAB index numbers. One way to think this through is with a thought experiment: suppose there had been a target at range 20 km. Then the first echo sample would be in our first received signal sample. Determine at which sample the peak would occur if you match filtered such a signal; that sample should be labeled as 20 km range in this example.

Now repeat this process using the LFM pulse *x*[*n*]. Place the two target echoes at the same ranges as before and match filter the resulting data with an appropriate matched filter, but including time-domain Hamming weighting for range sidelobe control. Overlay a plot of the result on the same plot as the simple pulse output.

Are the two targets resolved in the simple pulse case? Are they resolved in the LFM case? In the LFM case, are the target peaks at the expected ranges? (They should be exactly at the expected ranges, not just “close”. If not, reconsider how you label the matched filter output axis.) What is the expected range resolution for each pulse, and how does that compare to the target spacing?

## Range-Doppler Coupling

In Section 4.6.4 we showed that the presence of an uncompensated Doppler shift of *FD* Hz on an up chirp waveform like that of Eqn. will cause a shift in the matched filter response peak of



Let *FD* = **/10; then *tpeak* = **/10, which is 10 **s for the chirp pulse we have been using. We can create this amount of range shift in the output of the matched filter by multiplying the received echo by the Doppler shift term exp(*j*2*FDt*). Inserting the values for *FD* and using the fact that our sampling interval is *Ts* = 1/*k* samples/second, where *k* is your oversampling factor of at least 10, the discrete time modulation becomes



and the Doppler-shifted echo waveform becomes



Create the sequence *y*[*n*] according to Eqn. and matched filter it using the filter matched to *x*[*n*], *i.e.* the matched filter for the echo with*out* the Doppler shift. Plot the matched filter output, labeling the axis in units of time in seconds. At what time does the peak of the output occur? Is this shifted by the expected amount from the peak when the filter and echo are perfectly matched? Because the “echo” and the filter no longer match perfectly, there should also be a reduction in the magnitude of the peak. What is the magnitude of the peak output you observe, and how does it compare to the correctly matched case?

Repeat the calculations above, but with a *down* chirp for the waveform. This can be obtained by simply complex conjugating your original upchirp. The matched filter impulse response must also be conjugated so that it matches the new waveform. However, the Doppler shift term *d*[*n*] is *not* conjugated; it is the result of the velocity of the presumed target, which we do not want to change. Where is the peak of the matched filter output for the down chirp, and how does that compare to the up chirp case? How might you use this result to make accurate range measurements despite unknown Doppler shifts? What would be the costs of your method?

# ADDITIONAL DATA AND PROCESSING ISSUES

## Ambiguity Function

While not required for this project, it is sometimes fun to compute and examine the ambiguity function of various waveforms. A MATLAB function that computes the ambiguity function, ambiguity.m, is included at the end of this assignment for your experimentation. The file is available at the class T-Square site in the “Extra Goodies” section. Also available is a GUI-based MATLAB ambiguity function “calculator” from Prof. Nadav Levanon of Tel Aviv University. Hints on using this are available at [http://www.eng.tau.ac.il/~nadav/amb−func.html](http://www.eng.tau.ac.il/~nadav/amb-func.html), and a document is also available that has some example calls and the plots that result. Prof. Levanon’s ambiguity calculator MATLAB code and the pdf file of examples are both included in the zip file NLevanon\_ambfn.zip on the T‑Square site.

**Useful MATLAB Functions**

Files containing the following MATLAB functions can be downloaded from the ECE6272 T-Square site.

**git\_chirp.m**

function x = git\_chirp( T, W, p )

%CHIRP generate a sampled chirp signal

% X = git\_chirp( T, W, <P> )

% X: N=pTW samples of a "chirp" signal

% exp(j(W/T)pi\*t^2) -T/2 <= t < +T/2

% T: time duration from -T/2 to +T/2

% W: swept bandwidth from -W/2 to +W/2

% optional:

% P: samples at P times the Nyquist rate (W)

% i.e., sampling interval is 1/(PW)

% default is P = 1

%

if nargin < 3

p = 1;

end

J = sqrt(-1);

%--------------

delta\_t = 1/(p\*W);

N = round( p\*T\*W ); %--same as T/delta\_t

nn = [0:N-1]';

x = exp( J\*pi\*W/T \* (delta\_t\*nn - (N-1)/2/p/W).^2 ); % symmetric version

**ambiguity.m**

function AF=ambiguity(x,N,M,Tf,plots,labels)

%

% ambiguity

%

% calling sequence: AF=ambiguity(x,N,M,Tf,labels)

%

% M-file to compute the ambiguity function of a real or

% complex vector

%

% x = complex input vector, the signal to be analyzed

% N = length of x

% M = fft size. M >= 2\*N-1.

% Tf = fast time sampling interval in seconds

% plots = optional argument to determine plots generated

% If plots = 'cuts', onle zero-delay and zero-Doppler plots

% are drawn. Anything else, and contour and mesh plots are

% also drawn.

% labels = optional argument to control axis labels.

% If labels=='normalize', axes are plotted relative to pulse

% length. If labels is any other value, absolute (seconds and Hz)

% labels are used.

%

% Mark Richards

% March 2002

% Revised August 2004

%

% ensure that data is used in row vector form. Be careful

% NOT to conjugate it at the same time it is transposed

% (MATLAB's transpose is a Hermitian)

[r c] = size(x);

if (c == 1)

x = x.';

end

% ensure adequate FFT size is used

if M < 2\*N-1

disp('Error in ambiguity.m: FFT size too small. Must have M >= 2\*N-1.')

return

end

% determine whether to do all 4 plots

surface\_plots = 0;

if (nargin < 5)

surface\_plots = true;

else

surface\_plots = ~strcmp(plots,'cuts');

end

% determine whether to use absolute or relative labeling

abs\_label = 0;

if (nargin < 6)

abs\_label = true;

else

abs\_label = ~strcmp(labels,'normalize');

end

% form matrix of overlapped signals. Deliberately include one sample in

% each direction in time that will be zero; makes for better plots

AF = zeros(2\*N+1,M);

for n=-N+2:N

ll = max(1,n);

ul = min(N,n+N);

if (n <= 1)

AF(n+N,1:n+N-1) = x(1:n+N-1).\*conj(x(2-n:N));

else

AF(n+N,n:N) = x(n:N).\*conj(x(1:N-n+1));

end

end

% now transform each column separately, and shift to get

% frequency origin in the middle of the array

for n = 1:2\*N+1

AF(n,:) = fftshift(M\*ifft(AF(n,:)));

end

% ambiguity function is magnitude (not squared) of result;

AF = abs(AF);

peak=max(max(AF));

% set up normalized and unnormalized time and frequency scales

f\_abs = ((-M/2:M/2-1)/M)\*(1/Tf);

t\_abs = (-N:N)\*Tf;

f\_rel = ((-M/2:M/2-1)/M)\*(1/Tf)\*(N\*Tf);

t\_rel = (-N:N)/N;

% do plots

if (surface\_plots)

figure(1);

if (abs\_label)

surf(f\_abs,t\_abs,AF);

axis([min(f\_abs),max(f\_abs),min(t\_abs),max(t\_abs),0,max(max(AF))])

xlabel('Doppler shift (Hz)')

ylabel('delay (sec)')

else

surf(f\_rel,t\_rel,AF);

axis([min(f\_rel),max(f\_rel),-1,1,0,max(max(AF))])

xlabel('normalized Doppler shift')

ylabel('normalized delay')

end

colormap('hsv')

title('Ambiguity Surface')

figure(2);

axis('square')

% c=[0.994\*peak,0.708\*peak,0.316\*peak,0.1\*peak];

c = [0.994\*peak, 0.708\*peak, 0.316\*peak, 0.1259\*peak, 0.1\*peak];

if (abs\_label)

contour(f\_abs,t\_abs,AF,c);

axis([min(f\_abs),max(f\_abs),min(t\_abs),max(t\_abs)])

xlabel('Doppler shift (Hz)')

ylabel('delay (sec)')

else

contour(f\_rel,t\_rel,AF,c);

axis([min(f\_rel),max(f\_rel),-1,1,])

xlabel('normalized Doppler shift')

ylabel('normalized delay')

end

title('-0.5, -3, -10, and -20 dB Contours')

end % end of "if (surface\_plots)" condition

figure(3);

axis('normal')

if (abs\_label)

plot(t\_abs,AF(:,M/2+1))

axis([min(t\_abs),max(t\_abs),0,max(AF(:,M/2+1))])

xlabel('delay (sec)')

ylabel('amplitude')

else

plot(t\_rel,AF(:,M/2+1))

axis([-1,1,0,max(AF(:,M/2+1))])

xlabel('normalized delay')

ylabel('amplitude')

end

title('Zero-Doppler Cut')

figure(4);

if (abs\_label)

plot(f\_abs,AF(N+1,:))

axis([min(f\_abs),max(f\_abs),0,max(AF(N+1,:))])

xlabel('Doppler shift (Hz)')

ylabel('amplitude')

else

plot(f\_rel,AF(N+1,:))

axis([min(f\_rel),max(f\_rel),0,max(AF(N+1,:))])

xlabel('normalized Doppler shift')

ylabel('amplitude')

end

title('Zero-Delay Cut')

1. Office: Klaus 3354, 404-894-2714, <mark.richards@ece.gatech.edu>. Office hours TBD, but drop-ins and appointments welcome. [↑](#footnote-ref-1)
2. The matched filter output for the complex-valued chirp waveform will be complex-valued also, although the value at the peak should be purely real, at least to within numerical roundoff error in MATLAB. [↑](#footnote-ref-2)
3. We also know that matched filter response to an LFM waveform is not exactly a sinc, due to the  terms that appear in the expression for the zero-Doppler cut of the ambiguity function (see Eqn. (4.97) in the text); but if the time-bandwidth product ** is reasonably large, as it is here, the first zero will be very close to 1/**. [↑](#footnote-ref-3)
4. This formula will be derived in Chapter 5 of the text in connection with Doppler processing. [↑](#footnote-ref-4)