GEORGIA INSTITUTE OF TECHNOLOGY

SCHOOL OF ELECTRICAL ENGINEERING

ECE 6272 FALL 2010

COMPUTER PROJECT #2

**SOLUTION NOTES**

# Effect of Time-Bandwidth Product on the LFM Chirp Spectrum

We think of the LFM chirp as having a spectrum that is a good approximation to a rectangle function. This is only true if the BT (time-bandwidth) product ** (** = swept instantaneous bandwidth, ** = pulse length) is reasonably high.

Figure 1 shows the spectrum of three LFM chirps, generated using the threechirps.m M-file in MATLAB (included at the end of these notes). We denote the three chirps as *xi*[*n*], *i*=1,2,3. In all three cases, the duration is 100 **s. The swept bandwidths are *i* = 100 kHz, 1 MHz, and 10 MHz, giving BT products of 10, 100, and 1000. The data is oversampled by 20% (oversampling factor *k* of 1.2) in each case to make the spectral details clearer while still having the spectrum occupy most of the plot. Furthermore, in each case I used an FFT size of *K* = 1000; this gives me good definition of the underlying discrete-time Fourier transform (DTFT) by providing a dense set of samples along the frequency axis.



Figure . The spectra of three LFM chirps.

The figure clearly shows that the chirp with a BT product of only 10 has a spectrum that is a very poor approximation to a rectangle. The BT product of 1000 gives a rather good approximation to a rectangular spectral magnitude, and the BT product of 100 is also fairly good. In practice, a BT product of 100 or better is usually considered large enough to model the LFM spectrum as a rectangle.

The next several paragraphs discuss how I normalized the three signals so that their spectrum amplitudes were approximately equal.

Since the chirp bandwidths differ, the sampling rate for each chirp is different. In particular, the sampling rates and corresponding sampling intervals *Ti* are



Since each pulse shares a duration of 100 **s, this also means that the number of samples in each pulse is different and in fact is proportional to the bandwidth:



Finally, we can compute the approximate bandwidth of each signal in normalized frequency units. The individual bandwidths in “analog frequency” are *i* Hz, so that the spectra are nominally rectangle functions extending from –*i*/2 Hz to +*i*/2 Hz. Analog frequencies are mapped to normalized “digital frequencies” according to the relation



Combining Eqns. and , we see that the cutoff frequencies ±*i*/2 therefore map to the normalized frequencies



Note that the cutoff frequency in normalized frequency is the same for all three chirps; this occurs because of their differing sampling frequencies. This is why their DFTs all have the same nominal cutoff frequency when plotted together in Figure 1, despite the differing analog bandwidths.

Since each chirp has the same amplitude of 1, it then follows that the energy *Ei* in the chirps is also proportional to the bandwidth:



where the last step follows from Eq. . We also know from Parseval’s theorem that the energy can be expressed in terms of the signal spectrum as follows:



Using Eqns. and and the Principle of Stationary Phase (PSP) approximation that *Xi*(**) is approximately a rectangle function of some amplitude *Ai* on the frequency interval  converts Eqn. to



Since all of our waveforms have the same value of *k*, **, and time-domain amplitude (namely, 1), Equation shows that their spectra will have different amplitudes proportional to the square root of their respective bandwidths Thus, if we normalize each chirp by the square root of its bandwidth, the normalized spectra amplitudes  will be comparable,[[1]](#footnote-1) which is convenient for plotting.

# Comparison of LFM and Simple Pulse Waveforms and Matched Filters

We now compare an LFM with a BT product of 100 with a simple pulse of the same duration and sampling rate. The M-file waveform.m, included at the end of these notes, is used for this and most of the remaining portions of the project. Both waveforms have a duration of ** = 100 **s; the LFM chirp has a bandwidth of ** = 1 MHz to give a BT product of 100. The 3-dB bandwidth of the simple pulse is 0.89/**= 8.9 kHz; the Rayleigh (peak-to-first null) bandwidth is 1/** = 100 kHz. In this and most of the remaining portions of the project I used an oversampling factor of *k* = 10. Given the BT product of 100, this means that my chirp waveform has 1000 samples. Consequently, I also choose 1000 samples as the length of my simple pulse.

We start by constructing the matched filter for each pulse, and computing the response of this filter to the echo from a single point target. We know from the text that this is simply the autocorrelation of the respective waveforms. The result is shown in Figure 2 on a linear scale, and in Figure 3 on a dB scale. The insets in Figures 2 and 3 expand some of the fine detail of the LFM matched filter sidelobes.

The difference in shapes is dramatic, but note that the peak value of both outputs is exactly 1000. Since both pulses have a duration of 1000 samples and an amplitude of 1, the energy in each pulse is , and the peak of the signal component at the matched filter output always equals the signal energy. Also note that the duration appears to be exactly 2** = 200 **s, as would be expected for the autocorrelation of a function that is ** = 100 **s seconds long. In terms of samples, the original waveform was *N* = 1000 samples long, while the autocorrelation is 1,999 samples (2*N*−1) long.



Figure . Comparison of respective matched filter outputs to chirp and simple pulses of the same length.



Figure . Same as , but on decibel scale.

The LFM chirp is called a “pulse compression” waveform because most of its energy at the output of the matched filter is compressed into a much narrower spike than is the case for the simple pulse of the same length. Expanding the plot around the mainlobe region, as shown in below, shows that the peak-to-null (Rayleigh) width of the response is almost exactly 1 **s, which is what we would expect for a sinc function corresponding to a rectangular spectrum of bandwidth 1 MHz. We must remember that the LFM waveform is not perfectly rectangular and therefore that the matched filter output shape is not exactly a sinc function; in fact, we know it has  terms in both the argument of the sinc function and multiplying the overall sinc function. The former tends to spread the mainlobe, while the latter tends to drive it to zero faster. However, we can use Eqn. (4.98) or (4.99) from the text for the case where the time-bandwidth product is 100 to find that the zero is expected to occur at 0.99 **s. This is in close agreement with our expectations for sinc functions corresponding to a 1 MHz rectangular spectrum, emphasizing again that for a time-bandwidth product of 100, the LFM spectrum is very reasonably modeled as a rectangle.



*Figure 4. Expansion of the mainlobe region of the LFM matched filter output.*

We also see from close examination of the sidelobes, such as in the inset on Figure 3, that the first sidelobe is about 13.4 dB down from the peak, again closely matching (especially in light of the extra  weighting) the 13.2 dB expected for a sinc function.

In contrast, the rectangular pulse matched filter output has its first zero at its edge, which occurs at 100 **s. Its peak-to-null width is therefore 100 **s, 100 times the peak-to-null width of the LFM output. This is to be expected because of the 100:1 difference in BT product of the two waveforms.

# Sidelobe Suppression

We can see from that the peak sidelobe in the LFM matched filter output is approximately 13.4 dB below the peak. This should not be surprising; the spectrum is, by design, approximately a rectangle function so that its inverse Fourier transform is approximately a sinc function, which would have –13.2 dB peak sidelobes. On the other hand, the spectrum is not an exact rectangle, so we do not expect exactly –13.2 dB peak sidelobes.

In many, perhaps most, applications, –13 dB sidelobes are not acceptable and must be reduced. The most common sidelobe suppression technique is to apply a window function. In FIR filter design, we apply the window to the filter impulse response in the time domain in order to reduce the filter sidelobes (stopband response) in the frequency domain. That is, if we want to suppress sidelobes in one domain, we apply the window in its complementary Fourier domain. Applying this thinking to the LFM matched filter, we want to suppress sidelobes in the time (range) domain, therefore we should apply the window in the frequency domain.

One way to do this would be to perform conventional matched filtering in the fast time domain; compute the DFT of the matched filter output; apply the window to that spectrum; and then take the inverse DFT to obtain the reduced-sidelobe response. However, by performing the matched filtering in the frequency domain, we can combine the matched filter operation with the windowing to reduce the required computation. A block diagram of one way to arrange the flow of operations is shown in Figure 5. This diagram shows the window applied to the filter frequency response before the actual filtering step. This has the advantage of allowing the windowing to be done off-line, *i.e.* not in real time; one simply computes the product of the window and the matched filter frequency response once, and then uses it to filter the real-time data. We could just as well apply the window to the product of the signal and matched filter frequency responses. The end result at the output would be identical, but the windowing would have to be repeated every time there was a new signal to filter, and in real time.



Figure . Flow diagram for LFM sidelobe suppression by frequency domain windowing.

Figure 6 illustrates the application of a Hamming window function to the LFM chirp oversampled by 1.2x.[[2]](#footnote-2) The DFT of the chirp has been rotated to place the zero frequency point in the middle of the plot. Note that the window function has been aligned so that its center is aligned with the center of the LFM spectrum. Furthermore, the width of the window corresponds to the nominal width of the LFM spectrum, namely ±**/2 Hz.



Figure . Hamming window function aligned with respect to chirp spectrum and cut off at ±/2 Hz.

Figure 7 shows the output obtained by applying the operations shown in Figure 5 to the echo from a single point scatterer, that is, to a replica of the transmitted waveform. An oversampling rate of 10x was used. The results are shown with (blue) and without (green) Hamming weighting.

The windowed filter response suffers a loss in the amplitude of the peak. This is inevitable because the window modifies the matched filter frequency response, so that it is no longer exactly matched to the transmitted waveform. Consequently, there must be a loss in peak response. In Figure 7 the peak is reduced from 60 dB to 54.65 dB, a loss of 5.35 dB. The predicted loss is given by



A 16,384 point FFT was used for the DFTs in my solution to get good detail in the spectrum. The size of the Hamming window is then (16,384/10), which rounds to 1,638 samples. Evaluating the equation for *LPG* for a 1638-point Hamming window gives a predicted loss of 5.36 dB, in excellent agreement with the measured value of 5.35 dB.



Figure . Output of frequency domain matched filter, with and without Hamming weighting.

The primary purpose of windowing is reduction of sidelobes. In the unwindowed case, the peak occurs at 60 dB and the peak sidelobe is at 46.5 dB, which is 13.5 dB below the peak. In the windowed case, the peak is at about 54.6 dB, while the peak sidelobe (not the same as the first sidelobe in the weighted case) is at 17.5 dB, and is therefore down 37.1 dB. Thus, use of the Hamming weighting has improved peak sidelobe suppression by 23.6 dB (37.1 – 13.5).

The final effect of interest caused by the windowing is the broadening of the matched filter mainlobe, which represents a loss of range (time) resolution. Inspection of the location of the first zero of the response for each case shows it to occur at about 1 **s in the unwindowed case, and 1.93 **s in the windowed case.[[3]](#footnote-3) This latter value is a little less than the 2 **s we would normally anticipate due to the 2x broadening expected of a Hamming window.

There is a legitimate question as to whether the Hamming window should be chosen to cut off at ±**/2 Hz as shown in Figure 6. This choice is obviously motivated by the swept instantaneous frequency range of the chirp, but because of the modest BT product of 100, the spectrum does not cut off sharply at ±**/2. Some perhaps non-trivial energy outside of ±**/2 is zeroed by the window in this case. Figure 8 repeats the experiment of Figure 7, but with the Hamming window expanded by 10% in frequency to cover more of the LFM spectrum tails.



Figure . Same as , but with 10% expanded bandwidth Hamming window.

Close inspection of Figure 8 shows that the peak is now reduced only to about 55.4 dB. The corresponding loss is 4.6 dB instead of the previous 5.35 dB. Thus, the loss has been reduced (improved) by 1.34 dB by not discarding the energy at the tails of the spectrum. On the other hand, the peak sidelobe (which is now also the first sidelobe) is about 32.5 dB down from the peak, not quite as good as the 37.2 dB for the case where the Hamming window cutoff was at ±**/2. As the BT product gets larger, the cutoff of the signal spectrum becomes sharper, so that for large BT products, one should most likely cutoff the window at ±**/2 Hz.

Finally, we try time-domain weighting of the receiver impulse response. The textbook (Section 4.6.2) showed, using the principle of stationary phase, that an LFM pulse with a time-domain amplitude *A*(*t*) would have a spectrum whose magnitude followed the same shape as *A*(*t*), but spread over the frequency range ±**/2 Hz. Since our goal is to achieve a filter impulse response whose spectrum has a Hamming shape, this means that applying the Hamming weight to the filter impulse response should achieve that result. The output of the resulting filter, shown in Figure 9, has the same general character as the frequency-domain weighting result of Figure 7, but with some differences in details of the sidelobe structure. The peak is reduced from 60 dB to 54.64 dB with weighting, a reduction of 5.36 dB that agrees with the predicted value. The peak sidelobe of the weighted response (which is the first sidelobe in this case) is 40.7 dB below the mainlobe peak, 3.5 dB better than my first frequency-domain case. The Rayleigh width of the unwindowed case remains 1 **s, while the windowed case Rayleigh width is 1.97 **s; closer to the 2 **s expected for the Rayleigh window than the 1.93 **s we observed in the frequency domain weighted case.

For convenience, Figure 10 plots the frequency- (blue curve) and time-domain (green curve) weighted responses on the same plot. The main part of the figure plots lines connecting the sidelobe peaks to enable an easy comparison of the sidelobe levels in the two cases. The inset is the full response. The difference in sidelobe levels ranges from about 2.5 to as much as about 5 dB.



Figure . Matched filter output with and without time-domain weighting of the filter impulse response. Compare to Figure 7.

 

Figure . Comparison of sidelobe levels of frequency-domain weighted (blue) and time-domain weighted (green) matched filter outputs.

# All-Range Matched Filtering and Two-Target Resolution

This portion of my sample is represented by the program LFM\_twotargets.m included at the end of these notes. We set up simple and LFM pulses of length 100 **s and amplitude 1. The swept bandwidth of the LFM pulse is ** = 1 MHz. The oversampling factor is *k* = 2, so the sampling rate is 2 Msamples/sec. The corresponding sampling interval in fast time is 0.5 **s, equivalent to 75 m between range samples. Each pulse is 200 samples long.

According to the problem specification, the first sample of the data is taken at a time delay corresponding to 20 km range; this is the range we should associate with sample #1 in the simulated data. The first target echo appears at sample #101; the corresponding range is therefore 20 km + (100 samples)(75 m/sample) = 27.5 km. The second target at sample #141 must be at range 30.5 km. Because the two target responses are only 100 samples apart, but the pulses are 200 samples long, the echoes from the two targets will overlap considerably.

Figure 11 shows the output of the two matched filters. For the simple pulse case (blue), the two targets are not resolved. This is to be expected; the range resolution is *c*/2 = 15 km, but the targets are separated by only 3 km, so they are not resolved in range. For the LFM case (green), however, they are easily resolved, as seen by the two distinct peaks. The LFM range resolution is *c*/2** = 150 m, much finer than the 3 km spacing.



Figure . Matched filter output for simple pulse (blue) and LFM pulse with Hamming weighting (green) in the two-target case. The inset shows detail of the LFM output around the peaks.

Note that the LFM peaks are lower amplitude than the unresolved simple pulse peak, despite the two types of pulses having the same energy. There are two contributing factors. The first is that in the simple pulse case, the two triangular outputs for each target are overlapping, adding in phase, and resulting in a peak value greater than the single pulse energy of 200. The second is that the LFM pulse peaks are reduced from 200 due to the Hamming weighting as discussed earlier.

The inset in the figure shows additional detail around the LFM peaks. Note that the two peaks occur exactly at the correct ranges of 27.5 and 30.5 km. However, if you examine your output data from the convolution operation, you will find that the peaks do not occur at samples 101 and 141 that correspond to these ranges as discussed above, but rather at samples 300 and 340. This shift is due to the delay of the matched filter, which is *N*−1 samples (199 in this case). One way to see this is to simply do the autocorrelation of one of the pulse waveforms and observe the sample number at which the peak occurs. This delay must be accounted for in labeling the range axis of the matched filter outputs. To be explicit, the first LFM peak occurs at sample #300. To convert this to range, subtract the filter delay of 199, giving sample #101. This is 100 samples after sample #1, which corresponds to 20 km. Since the samples are 75 m apart, the range of the peak at sample 300 is 20 km + (300−199−1)(75) = 27.5 km.

Note that we implicitly assumed that the two targets both had reflectivities with the same phase, *i.e.* the second target did not add an additional phase shift with respect to the first. However, if we are at X band (10 GHz) for example, the wavelength is 3 cm. Since received phase is shifted by  radians, a range change for one scatterer of just **/4 (7.5 mm) within a range bin of 150 m for the LFM case and 15 km for the simple pulse case is enough to change the phase of one of the echoes by 180°, so that overlapping responses now subtract instead of adding. This will make little difference to the LFM echoes if they are resolved, but it can have a profound effect when the main responses overlap significantly, as the simple pulse outputs do. shows the effect of a 180º phase shift on the second of the two echoes; the inset provides additional detail around the center of the target region. The change in the resolved LFM outputs is not readily visible (there is a small shift in peak magnitudes and in the details of the sidelobe responses), but the effect on the highly overlapped simple pulse outputs is dramatic; in fact, the two scatterers are now resolvable due to the notch at *t* = 0. (However, it is not clear what the locations of the two scatterers are exactly, since there is no single peak for each one.) One cannot rely on a lucky relative phase to resolve scatterers. To obtain consistent resolution, a waveform with adequate bandwidth must be used, so that the matched filter output peak is inherently narrow enough to provide the desired resolution no matter what the details of the relative phase of the scatterers.



Figure . Output of matched filter for the same two pulses and two targets places 1.5 km (10 s) apart in range.

# Range-Doppler Coupling

In the last experiment, unweighted time domain matched filtering was applied to a signal consisting of the same up chirp used above, but with a Doppler shift corresponding to 10% of the swept bandwidth (thus, 100 kHz) imposed on the returned echo. This is a large Doppler shift with our 1 MHz bandwidth at common radar frequencies, but is convenient for clarity of the displays below.

Because of the range-Doppler coupling phenomenon, we expect the Doppler shift to cause the peak of the matched filter output response to be shifted away from its correct location. The predicted shift in the peak location is



where we have used the particular values ** = 100 **sin the last step of the equation above. We thus expect the matched filter output peak to occur 10 **s early. The leftmost peak in Figure 13 is the response of the matched filter for the up chirp waveform and the specified Doppler shift. The peak occurs at –10 **s as predicted.



Figure . Output of matched filters to the same Doppler shift, but measured with complementary up and down chirps.

The results plotted in Figure 13 suggest a way to compensate for the erroneous range measurements (peak locations) that are caused by uncompensated Doppler shifts on the received signal. At least for isolated targets, we could transmit an up chirp on one pulse, and the corresponding down chirp on the next pulse. The true range location would then be the average of the two measured peak locations, since they will be displaced from the correct location by equal amounts but in opposite directions.

The second (rightmost) peak in Figure 13 is the result when we use a *down*chirp with the same swept bandwidth and duration as the original upchirp; we simply reverse the direction of the frequency sweep. The same positive Doppler shift is applied to the waveform, and the peak now occurs at +10 **s.

Because the filter is no longer matched to the received waveform (due to the extra Doppler term), we should expect a reduction in peak amplitude as well. In fact, we can see that the peak is reduced from a value of 1000 in Figure 2 to 900 in Figure 13.

Listing of threechirps.m

% threechirps

%

% M-file for ECE6272 computer project #2 on waveforms

%

% Mark A. Richards, Feb. 2000

% set up pulse length and oversampling rate

T = 100e-6; % pulse length (sec)

OS = 1.2; % chirp oversampling factor

% use supplied chirp function to create three complex chirps of

% specified bandwidth and length and oversample ratio

% note that each signal will be of a different length because the

% bandwidths are not the same, therefore neither are the sampling rates

xc10 = git\_chirp(T,10/T,OS); % BT=10

xc100 = git\_chirp(T,100/T,OS); % BT=100

xc1000 = git\_chirp(T,1000/T,OS); % BT=1000

% power in each signal also is proportional to bandwidth, because

% sampling rate is proportional to bandwidth but time duration

% is constant. For "niceness" of the plot, normalize by square

% root to length so all spectra will have about the same amplitude.

xc10 = xc10/sqrt(length(xc10));

xc100 = xc100/sqrt(length(xc100));

xc1000 = xc1000/sqrt(length(xc1000));

% compute the spectra of all three, shifting the origin to the

% center of the plot, and plotting against normalized frequency

N=length(xc1000);

XC10 = abs(fftshift(fft(xc10,N)));

XC100 = abs(fftshift(fft(xc100,N)));

XC1000 = abs(fftshift(fft(xc1000,N)));

% define frequency variable and do the plot

freq = ((0:N-1)/N)-0.5;

plot(freq,[XC10 XC100 XC1000]);

xlabel('normalized frequency (cycles)'); ylabel('spectrum amplitude')

Listing of waveform.m

% waveform

%

% M-file for ECE6272 computer project #2 on waveforms

%

% Mark A. Richards, Feb. 2000

% updated Feb. 2002

% updated Sep. 2006

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% set up pulse length and sampling rate

T = input('Enter pulse length (sec): '); % 100 us pulse length

% use supplied chirp function to create complex chirp of

% specified bandwidth and length

B = input('Enter chirp swept bandwidth (Hz): ');

OS = input('Enter chirp oversampling factor: ');

disp(['Chirp BT product =',num2str(B\*T)]);

Fs = OS\*B; % fast time sampling rate

Ts = 1/Fs; % fast time sampling interval

xc = git\_chirp(T,B,OS);

N = length(xc);

% create simple pulse of same duration and sampling rate

xs=ones(N,1);

% do matched filter output (autocorrelation function) of each

% waveform and compare on same scale

xcc = xcorr(xc);

xss = xcorr(xs);

time = (-N+1:+N-1)\*Ts;

figure(1);

plot(time,abs([xss xcc])); xlabel('time (sec)'); ylabel('amplitude');

% normalize and repeat plot on log scale

xcc = xcc/max(abs(xcc));

xss = xss/max(abs(xss));

figure(2);

plot(time,20\*log10(abs([xss xcc])));

axis([min(time) max(time) -50 0]);

xlabel('time (sec)'); ylabel('amplitude (dB)');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Now we want to look at range resolution. Set up two scatterers

% 10 us apart; the received signals are then duplicates of the

% transmitted signal, overlapped with appropriate delay.

offset = round(10e-6/Ts);

rc = [xc;zeros(offset,1)] + [zeros(offset,1);xc];

rs = [xs;zeros(offset,1)] + [zeros(offset,1);xs];

% now do the match filtering by correlating with the respective

% reference pulse

rcc = xcorr(rc,xc);

rss = xcorr(rs,xs);

time = (-N-offset+1:+N+offset-1)\*Ts;

figure(3);

plot(time,abs([rss rcc])); xlabel('time (sec)'); ylabel('amplitude');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%

% Now we want to do sidelobe suppression on the chirp, so we switch to

% frequency domain matched filtering.

% compute the spectrum of the chirp signal and fftshift it to put

% the origin in the middle of the array (this shift is mainly

% for plotting convenience, it is not necessary). We use a 16384

% point fft to get good spectrum definition.

Nfft = 4\*4096;

Nf2 = Nfft/2;

XC = fftshift(fft(xc,Nfft));

freq = ((0:Nfft-1)-Nf2)/Nfft;

% compute the length of the spectral segment that contains

% most of the spectrum energy, namely the number of samples

% that cover a range of B Hz. Keep it integer. Generate the

% Hamming window and get it centered in an array the same size

% as the signal spectrum.

nspec = round(Nfft/OS);

h = hamming(nspec);

hpad=[zeros(Nf2-1-floor(nspec/2),1);h;zeros(Nf2+1-ceil(nspec/2),1)];

figure(4)

plot(freq,[abs(XC)/max(abs(XC)) hpad]);

xlabel('normalized frequency (cycles)'); ylabel('amplitude');

% apply the window to the matched filter frequency response

HC = hpad.\*conj(XC);

% Do the matched filter with and without weighting,

% inverse transform, and plot the result.

% Compare to the unwindowed time-domain matched filter.

% Plot on dB scale to facilitate looking at sidelobes.

% Also pad out and align time-domain unweighted matched filter

% for comparison plotting. Plot only for +/- T seconds.

YC = HC.\*XC;

yc = fftshift(ifft(fftshift(YC)));

ZC = conj(XC).\*XC;

zc = fftshift(ifft(fftshift(ZC)));

time = Ts\*(-Nf2:Nf2-1);

indexT = (abs(time)<=T);

figure(5);

plot(time(indexT),20\*log10(abs([yc(indexT) zc(indexT)])));

big = max(20\*log10(abs(zc))); axis([-T +T big-60 big]);

xlabel('time (sec)'); ylabel('amplitude (dB)')

% check loss in processing gain

LPG\_measured = 20\*log10(max(abs(yc))) - 20\*log10(max(abs(zc)))

LPG\_formula = 10\*log10((sum(h)/length(h))^2)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%

% Repeat the frequency domain sidelobe suppression on the chirp with a

% 10% increase in span of the window

% compute the spectrum of the chirp signal and fftshift it to put

% the origin in the middle of the array (this shift is mainly

% for plotting convenience, it is not necessary). We use a 16384

% point fft to get good spectrum definition.

Nfft = 4\*4096;

Nf2 = Nfft/2;

XC = fftshift(fft(xc,Nfft));

freq = ((0:Nfft-1)-Nf2)/Nfft;

% compute the length of the spectral segment that contains

% most of the spectrum energy, namely the number of samples

% that cover a range of B Hz. Keep it integer. Generate the

% Hamming window and get it centered in an array the same size

% as the signal spectrum.

nspec = round(1.1\*Nfft/OS);

h = hamming(nspec);

hpad=[zeros(Nf2-1-floor(nspec/2),1);h;zeros(Nf2+1-ceil(nspec/2),1)];

figure(6)

plot(freq,[abs(XC)/max(abs(XC)) hpad]);

xlabel('normalized frequency (cycles)'); ylabel('amplitude');

% apply the window to the matched filter frequency response

HC = hpad.\*conj(XC);

% Do the matched filter with and without weighting,

% inverse transform, and plot the result.

% Compare to the unwindowed time-domain matched filter.

% Plot on dB scale to facilitate looking at sidelobes.

% Also pad out and align time-domain unweighted matched filter

% for comparison plotting. Plot only for +/- T seconds.

YCP = HC.\*XC;

ycp = fftshift(ifft(fftshift(YCP)));

ZC = conj(XC).\*XC;

zc = fftshift(ifft(fftshift(ZC)));

time = Ts\*(-Nf2:Nf2-1);

indexT = (abs(time)<=T);

figure(7);

plot(time(indexT),20\*log10(abs([ycp(indexT) zc(indexT)])));

big = max(20\*log10(abs(zc))); axis([-T +T big-60 big]);

xlabel('time (sec)'); ylabel('amplitude (dB)')

% check loss in processing gain

LPG\_measured = 20\*log10(max(abs(ycp))) - 20\*log10(max(abs(zc)))

LPG\_formula = 10\*log10((sum(h)/length(h))^2)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%

% Now try time domain sidelobe suppression on the chirp. We go back to

% time domain matched filtering, since that is more convenient now

% do matched filter output (autocorrelation function) of waveform with and

% without hamming weighting

xcc = xcorr(xc);

xcw = xcorr(xc,hamming(N).\*xc);

time = (-N+1:+N-1)\*Ts;

figure(8);

plot(time,abs([xcw, xcc]));

xlabel('time (sec)'); ylabel('amplitude');

figure(9);

plot(time,20\*log10(abs([xcw, xcc])));

big = max(20\*log10(abs(xcc))); axis([-T +T big-60 big]);

xlabel('time (sec)'); ylabel('amplitude (dB)')

% check loss in processing gain

LPG\_measured = 20\*log10(max(abs(xcc))) - 20\*log10(max(abs(xcw)))

LPG\_formula = 10\*log10((sum(h)/length(h))^2)

% Do a plot comparing frequency- and time-domain weighted responses

figure(10);

yc = yc(indexT); yc = yc(2:end-1);

plot(time,20\*log10(abs([yc xcw])));

big = max(20\*log10(abs(zc))); axis([-T +T big-60 big]);

xlabel('time (sec)'); ylabel('amplitude (dB)')

% Do another plot to follow sidelobe level trends of the two

[pxcw txcw] = peaks(20\*log10(abs(xcw)));

[pyc tyc] = peaks(20\*log10(abs(yc)));

figure(11)

plot(time(pyc),20\*log10(yc(pyc))); hold on

plot(time(pxcw),20\*log10(xcw(pxcw)),'g'); hold off

big = max(20\*log10(abs(zc))); axis([-T +T big-60 big]);

xlabel('time (sec)'); ylabel('amplitude (dB)')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%range-Doppler coupling

%

% first create a Doppler shift term corresponding to an expected time

% shift due to range-Doppler coupling of 10 us

doppler=exp(j\*2\*pi/10/OS\*(0:N-1)');

% now create doppler-shifted response to our original up chirp and process

% through the matched filter for the original waveform

xc1 = doppler.\*xc;

xcc1 = xcorr(xc1,xc);

% to create a down chirp, all we need to do is conjugate the original

% up chirp; so repeat above using conj(xc)

xc2=doppler.\*conj(xc);

xcc2 = xcorr(xc2,conj(xc));

% plot the two responses on a common axis

time = (-N+1:+N-1)\*Ts;

figure(12);

plot(time,abs([xcc1 xcc2])); xlabel('time (sec)'); ylabel('amplitude');

Listing of LFM\_twotargets.m

% LFM two\_targets

%

% M-file for ECE6272 computer project #2 on waveforms

%

% Mark A. Richards, Sept. 2010

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% % set up pulse length and sampling rate

% T = input('Enter pulse length (sec): '); % 100 us pulse length

%

% % use supplied chirp function to create complex chirp of

% % specified bandwidth and length

% B = input('Enter chirp swept bandwidth (Hz): ');

% OS = input('Enter chirp oversampling factor: ');

clear all

T = 100e-6; B = 1e6; OS = 2; c = 3e8;

disp(['Chirp BT product =',num2str(B\*T)]);

Fs = OS\*B; % fast time sampling rate

Ts = 1/Fs; % fast time sampling interval

xc = git\_chirp(T,B,OS);

N = length(xc);

T1 = 101; T2 =141;

% create simple pulse of same duration and sampling rate

xs=ones(N,1);

% create two-target echo data for both types of pulse

ys = zeros(401,1);

ys(T1:T1+N-1) = xs;

ys(T2:T2+N-1) = ys(T2:T2+N-1) + xs;

yc = zeros(401,1);

yc(T1:T1+N-1) = xc;

yc(T2:T2+N-1) = yc(T2:T2+N-1) + xc;

% do matched filter output of each waveform by explicit convolution. In

% the LFM case, include Hamming window for range sideobe suppression

hs = xs;

zs = conv(ys,hs);

hc = conj(xc(end:-1:1)).\*hamming(N);

zc = conv(yc,hc);

range = (0:400+N-1)\*(c\*Ts/2) + 20e3 - (N-1)\*(c\*Ts/2);

figure(1);

plot(range,abs([zs zc])); xlabel('time (sec)'); ylabel('amplitude');

1. I say “comparable” instead of “equal” because the argument depends on modeling the spectrum as a rectangle, and the quality of this approximation varies significantly with the BT product. It is probably *not* a very good assumption for the ** = 10 case, but is much better for the ** = 100 and 1000 cases. [↑](#footnote-ref-1)
2. This low oversampling rate was used only for this figure, not for the actual matched filtering, so the spectra would occupy most of the plot range rather than being squeezed into the central 10% of the plot. [↑](#footnote-ref-2)
3. I actually increased the oversampling to 30x to get better definition of the first zero in the windowed case. [↑](#footnote-ref-3)