Fall 2021 EE 313 Linear Systems and Signals Prof. Evans

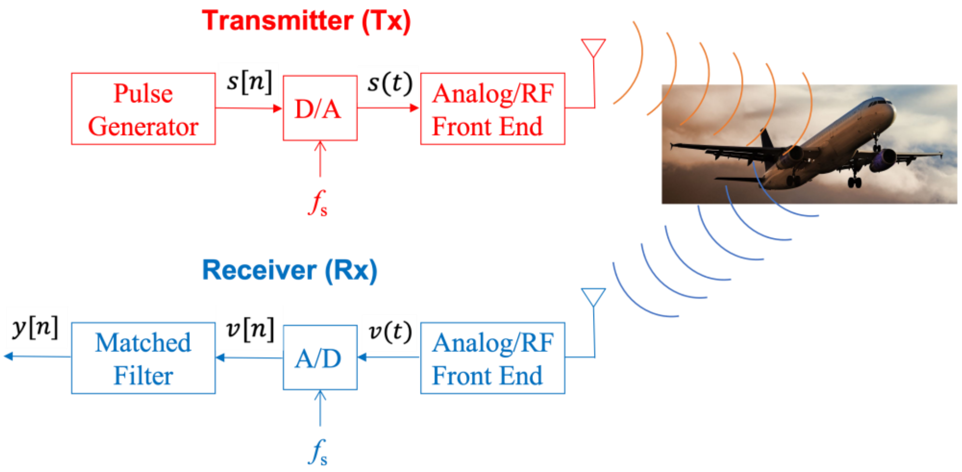
Mini-Project #2: Wireless Localization

Prof. Brian L. Evans and Mr. Firas Tabbara

November 14, 2021, *Version 2.0*

1.0 Introduction

Wireless localization of objects in an environment has applications in automotive, cellular, sonar, and ultrasound systems. Automotive systems fuse data from radar systems and cameras mounted on the vehicle to detect stationary and moving objects in the environment. In 6G cellular systems, basestations will use knowledge of the smart phone's location to improve the connection speed, reliability and coverage. By knowing the user equipment location, the basestation can know the wireless channel characteristics to tune the type and direction of transmission to the user equipment. Sonar and ultrasound use acoustic waves to detect location of objects underwater and biological structures in vivo without invasive procedures, respectively, in much the same way that automotive radar and 6G cellular systems use electromagnetic waves. Electromagnetic waves are produced by acceleration of electric charge or transition of electrons between energy levels in atoms [1], whereas acoustic waves are caused by mechanical vibration. This project will simulate a radar system that localizes an object in the environment using electromagnetic waves and signal processing [2] as shown in the diagram below.



2.0 Overview

Radar systems use electromagnetic waves to localize objects in an environment. A common radar technique is to transmit a short pulse and process the received signal to detect when the pulse returns. Using the estimated roundtrip time T_d, one can estimate the distance d to the object using the speed of propagation of the electromagnetic wave in the environment c (approximately the speed of light) via

$$d = \frac{1}{2}\, c\, T_d$$

The pulse is transmitted periodically to obtain periodic estimates of an object's location. The delay in sending the next pulse is long enough to allow the pulse to bounce off an object and travel back to the receiver. Knowing the maximum distance to an object, one can determine how often to send pulses.

This approach has two degrees of freedom: (1) pulse shape and (2) method to detect the pulse in the received signal. We would like to choose a pulse shape that is resilient to the impairments experienced by the electromagnetic signal as it propagates through the air. The next subsections describe wireless signal impairment, chirp signals as resilient pulse shapes, matched filtering for robust detection, and complex-valued signals.

2.1 Wireless Signal Impairments

A propagating electromagnetic waveform is reflected, scattered, and absorbed each time it impinges on an object in the environment, with metal objects providing strong reflections. Each path from the transmitter to the object and back to the receiver will incur a different gain (attenuation) and delay. The additive combination of paths from transmitter to receiver causes time delays and frequency distortion. The relative motion between the object and the radar system causes a Doppler shift in frequency [7]. In addition, the electronics in the radar analog/RF front ends add thermal noise as well as their own frequency distortion.

2.2 Chirp Signals As Resilient Pulse Shapes

A chirp signal is resilient to many impairments experienced by propagating signals including thermal noise, delays, and frequency distortion. Chirp signals are also resilient to Doppler shifts due to the relative motion between the transmitter and receiver [7]. Chirp signals can be used not only to localize an object, but also estimate its velocity (speed). [8]

In a chirp signal, as mentioned in Section 3-8 of [3], the principal frequency increases (or decreases) with time. Per the homework #2 solution set [4], chirp signals have a wide variety of applications:

> ***Active sonar systems***. The transmitter sends out a "ping" as sound in form of a chirp and then receives sound. The time elapsed between the transmission and reception of the chirp indicates the roundtrip time experienced by the signal after bouncing off an object in the water and returning to the receiver. By receiving sounds in different directions using multiple microphones, the sonar can build a map of the objects in the water.
>
> ***Bats*** use chirps for echolocation. Pipestrelle bats use chirps that sweep down from 70 to 45 kHz. [5]
>
> ***Test & Measurement***. When one measures the response of a system to different frequencies, a time-consuming approach is to input a single sinusoid, measure the output, and repeat using many different frequencies. Instead, inputting a chirp can allow the measurement to performed in one take.
>
> ***4G/5G cellular communication*** systems periodically send a Zadoff-Chu chirp sequence to synchronize the transmitter and receiver as well as measure the frequency distortion in the electromagnetic propagation from transmitter to receiver (channel estimation). The Zadoff-Chu is a complex-valued chirp signal. [6]

The radar system will use linear frequency modulated (FM) chirp signals to sweep a range of frequencies linearly with time. The chirp will be complex-valued (see Section 2.4 on Complex-Valued Signals).

2.3 Matched Filtering for Robust Detection

Matched filtering detects a known pulse shape in a signal by correlating the signal with the known pulse shape. A detection is successful when the absolute value of the correlation signal exceeds a threshold.

The matched filter gets its name from its impulse response $h(t)$ being matched to the known pulse shape $g(t)$ according to the following formula:

$$h(t) = C\, g^*(T - t)$$

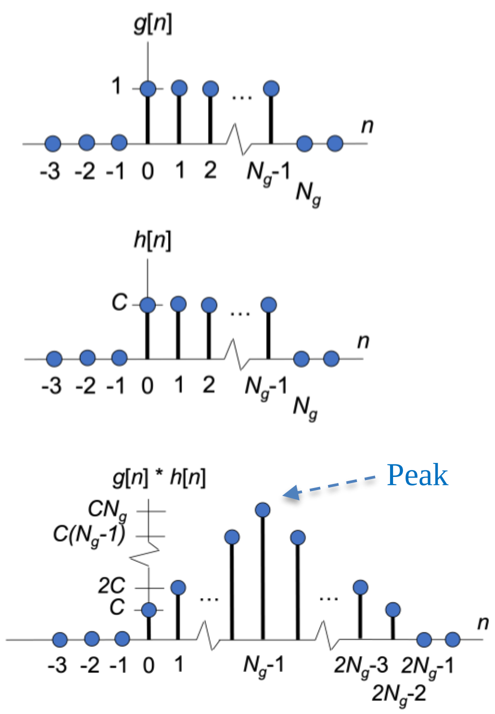
We form $h(t)$ by flipping $g(t)$ in time t, delaying by constant delay T, conjugating the amplitude, and scaling by non-zero constant C. T is often chosen to make the impulse response causal. The impulse response $h(t)$ will have the same duration as the known pulse shape $g(t)$. So, if the pulse shape is finite in duration, the matched filter will be a finite impulse response (FIR) filter.

In discrete time, the matched filter impulse response $h[n]$ is defined in terms of the pulse shape $g[n]$ as

$$h[n] = C\, g^*[N - n]$$

We form $h[n]$ by flipping $g[n]$ in time n, delaying by constant delay N, conjugating the amplitude, and scaling by non-zero constant C. N is often chosen to make the impulse response causal. The impulse response $h[n]$ will have the same duration as the known pulse shape $g[n]$. So, if the pulse shape is finite in duration, the matched filter will be an FIR filter.

Here's an example of how matched filtering works. Consider a causal rectangular pulse for $g[n]$ of length N_g samples and the corresponding matched filter impulse response $h[n]$ with $N = N_g$ shown on the right. When $C > 0$, the matched filter output when $g[n]$ is the input is a non-negative triangular pulse of $2\,N_g - 1$ samples whose peak at index $N_g - 1$ indicates a successful detection (below). When a negated pulse arrives, e.g. due to reflection, the result is a non-positive triangular pulse of $2\,N_g - 1$ samples whose valley at index $N_g - 1$ indicates a successful detection. (This is equivalent to C being negative in the plot of $g[n] * h[n]$ on the right.) This is why it is common to threshold against the absolute value of the matched filter output to determine a successful detection. Another common approach is to threshold against the absolute value squared of the matched filter output, which is the instantaneous power. The matched filter is actually designed to maximize the received transmitted power.



2.4 Complex-Valued Signals

Complex-valued chirp signals are used in radar and 4G/5G cellular systems, among others. Since in practice, a signal propagating through a medium is real-valued, we can transmit a complex-valued signal by transmitting its real and imaginary components on separate channels through the medium. In communication systems, the separate channels are called in-phase (I) and quadrature (Q). We can apply amplitude modulation using a cosine to a lowpass signal to generate the in-phase signal, and apply amplitude modulation by a sine to another lowpass signal to generate the quadrature signal. By subtracting the quadrature signal from the in-phase signal, we would have one real-valued signal to transmit and receive. Modulation by the cosine is orthogonal to modulation by sine, and hence, the IQ signals are in separate channels even though they are transmitted together. This is known as Quadrature Amplitude Modulation, which is used in Wi-Fi, cellular, and other communication systems.

3.0 Properties of the LFM Chirp Signal

The radar system being simulated uses a complex-valued linear frequency modulated chirp signal

$$s(t) = e^{j\pi\frac{W}{T}t^2} \text{ for } -\frac{T}{2} \le t \le \frac{T}{2}$$

The instantaneous frequency $f_i(t)$ is the derivative with respect to time t of the phase in radians divided by 2π to convert rad/s to Hz [2][8]

$$f_i(t) = \frac{1}{2\pi}\frac{d}{dt}\left(\pi\frac{W}{T}t^2\right) = \frac{1}{2\pi}\left(2\pi\frac{W}{T}t\right) = \frac{W}{T}t$$

Hence, the principal frequency varies linearly over $-\frac{W}{2} \le f \le \frac{W}{2}$ for $-\frac{T}{2} \le t \le \frac{T}{2}$.

3.1 Sampled Chirp Signal

(a) To aid in radar simulation, we create a discrete-time version $s[n]$ of the continuous-time chirp signal $s(t)$ by sampling at a rate $f_s = p\,W$ where p is the oversampling factor ($p \ge 1$):

$$s[n] = e^{j2\pi\alpha\left(n-\frac{N}{2}\right)^2} \text{ for } 0 \le n < N$$

Next, we determine formulas for α and N in terms p and time-bandwidth product TW. With sampling time T_s, the N samples span T seconds of continuous time, i.e. $T = N\,T_s$, from $-\frac{T}{2} \le t < \frac{T}{2}$. (Note that the endpoint at $\frac{T}{2}$ is not included when sampling.) With $f_s = pW$ and hence $T_s = \frac{1}{pW}$,

$$T = NT_s = \frac{N}{pW}$$

which gives

$$N = p\,T\,W \qquad \boxed{N = p\,T\,W}$$

We can find the formula for α by equating the first sample of $s[n]$ and $s(t)$:

$$s[0] = s\left(-\frac{T}{2}\right)$$

$$e^{j2\pi\alpha\left(-\frac{N}{2}\right)^2} = e^{j\pi\frac{W}{T}\left(-\frac{T}{2}\right)^2}$$

$$e^{j\pi\frac{\alpha N^2}{2}} = e^{j\pi\frac{TW}{4}} \qquad \boxed{\alpha = \frac{TW}{2\,N^2}}$$

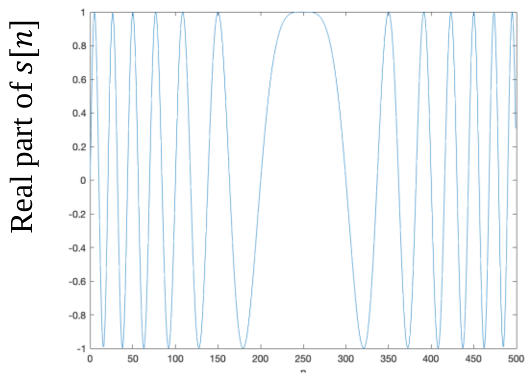
(b) We convert the above equations for α and N into a Matlab function to synthesize a discrete-time chirp that has two arguments p and TW and returns the complex-valued chirp signal $s[n]$:

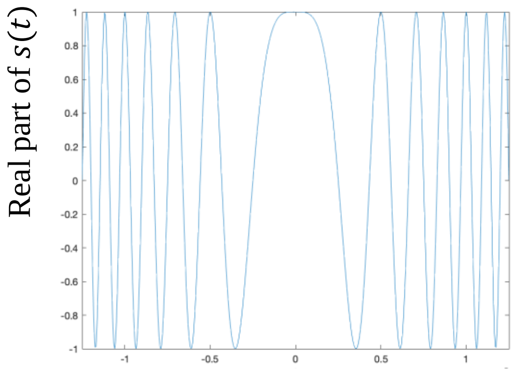
```
function s = dchirp( TW, p )
% DCHIRP   generate a sampled chirp signal
%   usage  s = dchirp( TW, p )
%          s : samples of a discrete-time "chirp" signal
%              exp(j pi (W/T) t^2 )   for -T/2 <= t <= T/2
%          TW : time-bandwidth product
%          p : sample at p times the Nyquist rate (W)
N = p*TW;
alpha = TW / (2*N^2);
n = 0 : N-1;
s = exp(j*2*pi*alpha*(n - N/2).^2);
```

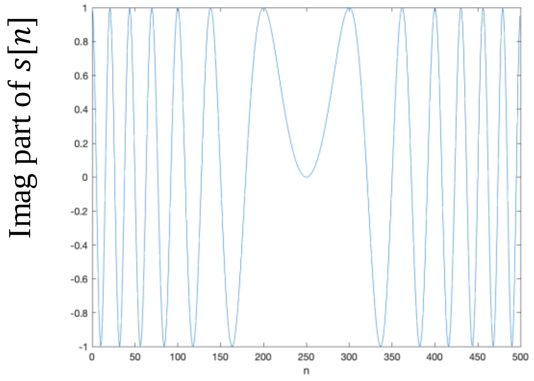
(c) Next, we generate a sampled chirp signal using the `dchirp` function using the radar parameters in Table 10.1 [8]. The parameters are repeated below along with the oversampling ratio $p = \frac{f_s}{W}$:

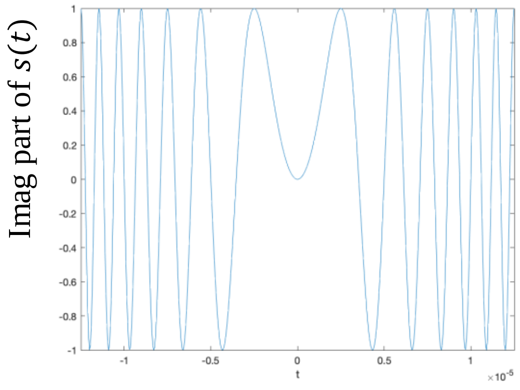
Parameter	***Symbol***	***Value***	***Units***
Pulse length	T	25	μs
Swept bandwidth	W	2	MHz
Sampling frequency	f_s	20	MHz
Time-bandwidth product	TW	50	dimensionless
Oversampling ratio	p	10	dimensionless

MATLAB code and plots of $s[n]$ and $s(t)$. We're plotting $s(t)$ as a sanity check on $s[n]$.







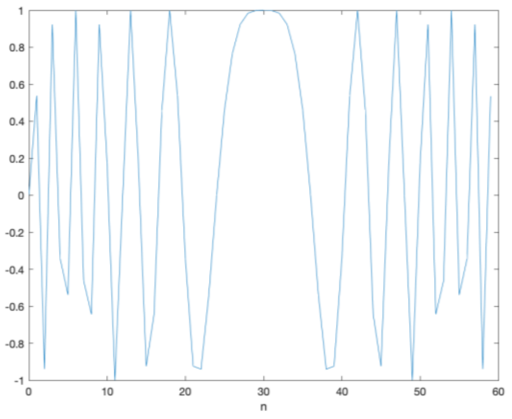


```
%%% Discrete-time chirp
TW = 50;
p = 10;
sofn = dchirp(TW, p);
N = length(sofn);
n = 0 : N-1;
figure;
plot(n, real(sofn));
xlabel('n');
figure;
plot(n, imag(sofn));
xlabel('n');
```

```
%%% Continuous-time chirp
T = 25E-6;     %% pulse length 25us
W = 2E6;       %% swept bandwidth 2MHz
fs = 20E6;     %% sampling rate 20 MHz
Ts = 1/fs;
t = (-T/2) : Ts : (T/2);
soft = exp(j*pi*W*(t.^2)/T);
figure;
plot(t, real(soft));
xlim([-T/2 T/2]);
xlabel('t');
figure;
plot(t, imag(soft));
xlim([-T/2 T/2]);
xlabel('t');
```

As continuous time increases from -*T*/2 to 0, the chirp decreases in frequency from -*W*/2 to 0, and increases in frequency from 0 to *W*/2 from time 0 to *T*/2. In discrete time, this corresponds to a decrease in frequency as discrete time increases from 0 to (*N*-1)/2 and an increase in frequency as discrete time increases from (*N*-1)/2 to (*N*-1).

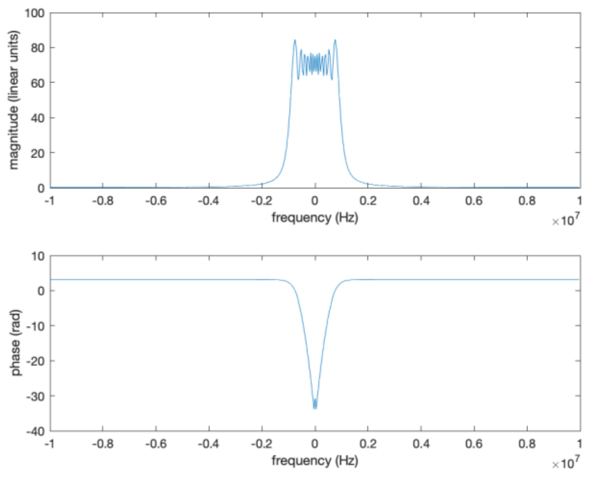
(d) A discrete-time chirp with oversampling ratio of $p = 1.2$ is plotted on the right. Near the endpoints, the chirp exhibits sinusoidal behavior. We'll plot the real part.



```
%%% Discrete-time chirp
TW = 50; p = 1.2;
sofn = dchirp(TW, p);
N = length(sofn);
n = 0 : N-1;
figure; plot(n, real(sofn));
xlabel('n');
```

3.2 Fourier Transform of a Chirp

The Fourier Transform measures the frequency components present on average in an interval of time. Even though the chirp is sweeping through a range of frequencies $-\frac{W}{2} \leq f \leq \frac{W}{2}$, the chirp signal has frequency content that is similar to a rectangular pulse over $-\frac{W}{2} \leq f \leq \frac{W}{2}$ (ideal lowpass signal). The similarity increases with increasing time-bandwidth product TW. We will use the fast Fourier transform (see Appendix A) to plot the magnitude and phase of the frequency components using `utplotspec.m` (see Appendix B). The chirp sweeps frequencies from -*W*/2 to *W*/2, i.e. from -1 MHz to 1 MHz according to the value for *W* in the table on page 5.



```
TW = 50; p = 10;
sofn = dchirp(TW, p);
fs = 20E6; Ts = 1/fs;
utplotspec(sofn, Ts);
```

4.0 Range Processing

4.1 Pulse-Compression Matched Filtering

(a) Convolution in time domain is multiplication in the frequency domain. The matched filter has a frequency response to cancel the phase of the frequency response of the pulse shape to maximize the magnitude squared (power) in the frequency domain. In the frequency domain, the matched filter frequency response is $\mathcal{H}(f) = S^*(f)$ or its equivalently its impulse response is

$$h(t) = s^*(-t) = e^{-j\pi\frac{W}{T}t^2}$$

The output of the matched filter is

$$y(t) = h(t) * s(t - T_d)$$

Equation (2-2) in [8] included a gain G in front of the $s(t - T_d)$ without explanation. The gain G likely represents attenuation experienced by the propagating waveform. For this subsection, we'll assume unity gain and no delay, i.e. $G = 1$ and $T_d = 0$.

To find the distance, a radar system will transmit a discrete-time complex chirp pulse signal $s[n]$ as a continuous-time signal through its antenna. The radar then correlates the received signal against the chirp signal using the matched filter. Based on the time at which the peak in the absolute value of the response occurs, we estimate the round-trip delay to find the distance of the object using

$$d = \frac{1}{2}\, c\, T_d$$

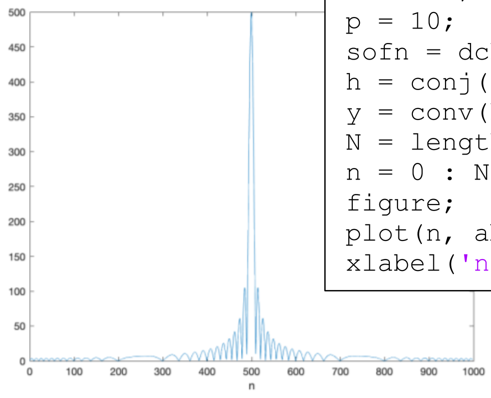
(b) The matched filter is a linear time-invariant (LTI) finite impulse response (FIR) filter whose impulse response is directly related to the pulse shape to be detected, as mentioned in Section 2.3. For a discrete-time complex-valued pulse shape $s[n]$, the matched filter would have an impulse response

$$h[n] = C\, s^*[N - n]$$

where N is a constant delay and C is a non-zero gain. For simplicity, we'll use $N = 0$ and $C = 1$:

$$h[n] = s^*[-n]$$

The relevant MATLAB commands are `conj` to conjugate the elements of the vector of signal values and `fliplr` to flip a vector of signal values. For a roundtrip delay of $T_d = 0$ seconds, the peak in the absolute value of the output of the matched filter occurs after the worst-case delay through the matched filter, which is the length of the filter of 500 samples. The peak value is equal to 500.



```
TW = 50;
p = 10;
sofn = dchirp(TW, p);
h = conj(fliplr(sofn));
y = conv(h, sofn);
N = length(y);
n = 0 : N-1;
figure;
plot(n, abs(y));
xlabel('n');
```

(c) Correlation measures similarity between two signals, with a higher absolute value meaning a higher similarity. The correlation $R_{xy}[k]$ between discrete-time signals $x[n]$ and $y[n]$ is defined as

$$R_{xy}[k] = \sum_{n=-\infty}^{\infty} x[n]\, y^*[n+k]$$

The outer discrete-time variable k indicates the shift in time between the two discrete-time signals. That is, correlation involves sliding $y^*[n+k]$ across $x[n]$ and summing the result for each value of the shift, k. Correlation is very similar to convolution:

$$R_{xy}[k] = \sum_{n=-\infty}^{\infty} x[n]\, y^*[n+k] = \sum_{n=-\infty}^{\infty} x[n]\, y^*[k+n]$$

In convolution, the $y^*[k+n]$ term would be $y^*[k-n]$. Note that the roles of k and n are reversed in correlation. We can write correlation as

$$R_{xy}[k] = x[n] * y^*[-n]$$

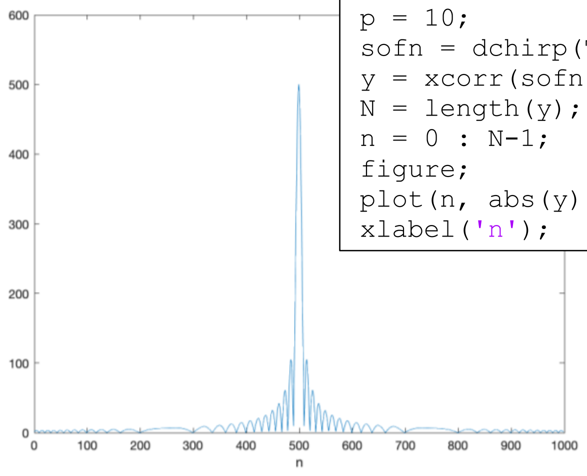
That is, prior to convolution, we'll flip $y^*[n]$ and convolution will flip it back. That way, we'll "fool" convolution to perform sliding $y^*[n]$ across $x[n]$.

Autocorrelation of signal $x[n]$ measures how similar a signal $x[n]$ is vs. shifts in time of itself:

$$R_{xx}[k] = \sum_{n=-\infty}^{\infty} x[n]\, x^*[k+n]$$

The maximum value for $R_{xx}[k]$ occurs when $k = 0$, i.e. when $x[n]$ is aligned with $x^*[n]$. For a complex-valued signal $x[n]$, the product $x[n]\, x^*[n]$ is the magnitude squared of $x[n]$ because the phases subtract out.

To verify our results, we use the `xcorr` function on MATLAB with one argument to perform the autocorrelation, and we plot its absolute value on the right. It is identical to the plot of the absolute value of the matched filter on the previous page.



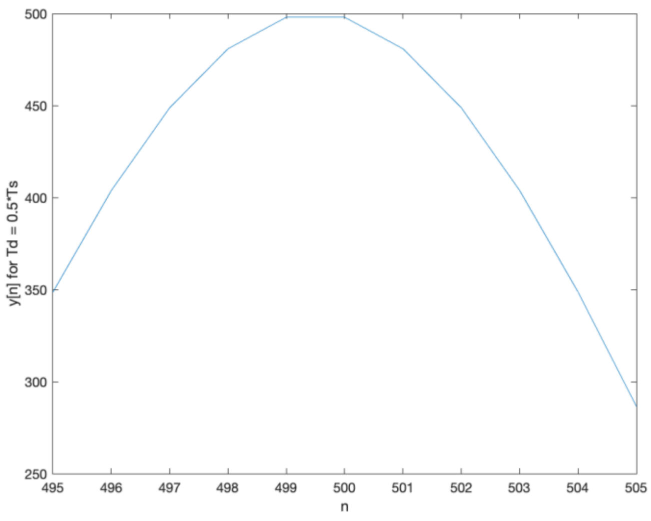
```
TW = 50;
p = 10;
sofn = dchirp(TW, p);
y = xcorr(sofn);
N = length(y);
n = 0 : N-1;
figure;
plot(n, abs(y));
xlabel('n');
```

4.2 Range Estimation

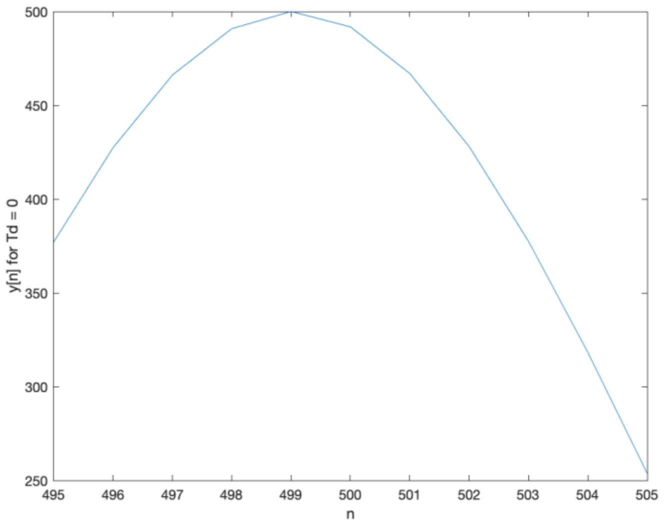
We estimate the range by estimating the round-trip time. We estimate the round-trip time by applying a matched filter to the received signal, picking the peak of the absolute value of the matched filter output, and subtracting the worst-delay of the matched filter from the time at which the peak occurred:

$$d = \frac{1}{2}\, c\, T_d$$

Because electromagnetic waves propagate in air at approximately the speed of light of $3 \times 10^8\ m/s$, error in the round-trip time T_d can lead to large errors in the range (distance) estimation for d.



When the matched filtering is performed in discrete time, the accuracy in the time at which the peak occurred in the sampling time, T_s. We can illustrate this by delaying the chirp signal by $\frac{1}{2}T_s$. The top plot shows the area around the peak of the absolute value of the matched filter output for $T_d = \frac{1}{2}T_s$ and the bottom plot shows the area for $T_d = 0$. When $T_d = 0$, a single peak occurs at index 499; however, for $T_d = \frac{1}{2}T_s$, two peaks occur at indices 499 and 500. The actual peak is at index 499.5, which is not on the integer grid.



In addition to having round-trip delays that are not integer multiples of the sampling time, the estimate of the round-trip time is also affected by multipath effects. That is, due to the wireless propagation along multiple paths from transmitter to receiver each with a different delay and gain, the absolute value of the matched filter output might have many peaks.

We can use post-processing of the points at which the absolute value in the matched filter output clears a threshold to improve the accuracy of the round-trip time estimate. For example, Exercise 2.3(d) in [8] suggests using a polynomial interpolator to obtain a high-resolution estimate of the round-trip time.

```
T = 25E-6;
W = 2E6;
fs = 20E6;
p = fs / W;
Ts = 1/fs;
Td = 0.5*Ts;
t = (-T/2) : Ts : (T/2);
sdelayed = exp(j*pi*W*((t-Td).^2)/T);
s = dchirp(T*W, p);
h = conj(fliplr(s));
y = conv(h, sdelayed);
N = length(y);
n = 0 : N-1;
figure;
plot(n, abs(y));
xlabel('n');
ylabel('y[n] for Td = 0.5*Ts');
xlim([495 505]);
```

5.0 Conclusion

Wireless localization has applications in automotive, cellular, sonar, and ultrasound systems. This project explored the use of radar signal processing to estimate the distance d to an object by estimating the round-trip time T_d and computing $d = \frac{1}{2}\, c\, T_d$. The accuracy in the distance depends heavily on the accuracy of the round-trip time.

This approach has two degrees of freedom: (1) pulse shape and (2) method to detect the pulse in the received signal.

A chirp pulse linearly sweeps a range of frequencies over an interval of time. A chirp is resilient to many impairments experienced by propagating signals including thermal noise, delays, frequency distortion, and Doppler shifts. [7] Chirp signals can be used not only to localize an object, but also estimate its velocity. [8]

Matched filtering is employed to detect a known pulse by correlating the received signal with the known pulse and processing the points in the absolute value of the output that exceed a threshold to estimate the round-trip time T_d. The simplest approach is to estimate T_d is to find the time at which the first peak occurs and subtract the worst-case delay through the matched filter. This method is limited in accuracy in the round-trip time by the sampling time and multipath effects. Although not explored in this project, polynomial interpolators can be applied to the points in the absolute value of the output that exceed a threshold to obtain high-resolution round-trip time estimates.

References

[1] T. Ryan, "What causes electromagnetic waves?", Socratic.org, accessed Oct. 24, 2021.
[2] B. L. Evans, "Mini-Project #2: Wireless Localization", EE 313 Linear Systems & Signals, UT Austin, Fall 2021.
[3] J. H. McClellan, R. W. Schafer & M. A. Yoder, *Signal Processing First*, 2003. Errata. On-line Companion.
[4] B. L. Evans, "Homework #2 Solution Set", EE 313 Linear Systems & Signals, UT Austin, Fall 2021.
[5] R. Mudhar, "Recordings of ultrasonic vocalisations of bats", accessed Oct. 24, 2021.
[6] "Zadoff-Chu Sequence", Wikipedia, accessed Oct. 24, 2021.
[7] "Doppler Effect", Wikipedia, accessed Oct. 24, 2021.
[8] C. S. Burrus, J. H. McClellan, A. W. Oppenheim, T. W. Parks, R. W. Schafer, H. W. Schuessler, "Chapter 10: Applications" (on Canvas), *Computer-Based Exercises for Signal Processing Using Matlab*, 1994. Code.
[9] "Radar Signal Characteristics", Wikipedia, accessed Oct. 24, 2021.
[10] Dina Katabi, "Lab 3: Wireless Localization", MIT 6.888 Wireless Communication Systems, Fall 2013.

Appendix A: Connections Between the Discrete-Time and Fast Fourier Transforms

Discrete-Time Fourier Transform (DTFT). The Discrete-Time Fourier Transform $X_{freq}(\omega)$ computes the frequency content in a discrete-time domain signal $x[n]$:

$$X_{freq}(\omega) = \sum_{n=-\infty}^{\infty} x[n]\, e^{-j\,\omega\, n}$$

An *infinite* number of coefficients could be needed to exactly represent $x(t)$.

Discrete-Time Fourier Series. A discrete-time periodic signal $x[n]$ with period N samples is composed of a constant term plus frequency components at integer multiples (harmonic) of a fundamental frequency of $2\pi/N$ rad/sample which is equivalent to f_s / N in Hz. Fourier series analysis computes the constant term $X[0]$ plus the magnitude and phase of each frequency term $X[k]$ for $k = 0, 1, \ldots, N-1$.

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]\, e^{j\left(k\frac{2\pi}{N}\right)n} \quad \text{where} \quad X[0] = \sum_{n=0}^{N-1} x[n] \quad \text{and} \quad X[k] = \sum_{n=0}^{N-1} x[n]\, e^{-j\left(k\frac{2\pi}{N}\right)n}$$

A *finite* number of Fourier series coefficients would always be needed to exactly represent $x[n]$.

Discrete Fourier Transform (DFT). The Discrete Fourier Transform (DFT) applies the Discrete-Time Fourier Series to a finite-length signal (or portion of a signal) to create a sampled version of the Discrete-Time Fourier Transform (DTFT):

$$X[k] = \sum_{n=0}^{N-1} x[n]\, e^{-j\left(\frac{2\pi}{N}k\right)n} = X_{freq}\left(\frac{2\pi}{N}k\right) \text{ for } k = 0, 1, \ldots, N-1$$

The DFT samples in both time and frequency. The DFT takes a vector of N discrete-time samples and produces a complex-valued vector of N frequency components.

Fast Fourier Transform (FFT) is a fast algorithm to compute the N discrete-time Fourier transform coefficients $X[k]$ from the N samples of a discrete-time signal $x[n]$. As with the continuous-time Fourier series, the discrete-time Fourier series assumes that the N samples of $x[n]$ represents the fundamental period of an infinitely long signal in the time domain. Unlike the continuous-time Fourier series, the discrete-time Fourier series always has a finite number of terms, N. One of the reasons for this is due to the Sampling Theorem, which says that the sampling rate $f_s > 2\, f_{\max}$ where $f_{\max}$ is the highest frequency of interest and hence $f_{\max} < ½\, f_s$. Sampling only captures continuous-time frequencies ($-½\, f_s$, $½\, f_s$) whereas continuous-time signals have frequencies. You can think of the discrete-time Fourier series as computing the Fourier series coefficients for frequency components from $-½\, f_s$ to $-½\, f_s$, which is a finite number because $f_s > 0$.

How fast is the FFT? The FFT to compute $X[k]$ above requires $2M \log_2 M$ real-valued multiplications and additions and $4M$ words of memory instead of the $4\, M^2$ and $M^2 + 4M$, respectively, for the direct matrix-vector implementation of the discrete-time Fourier series. A direct matrix-vector multiplication to compute $X[k]$ would create a complex-valued N x N matrix of the term $\exp(-j\ (2\pi/N)\ kn)$ for $k = 0, 1, \ldots, N-1$ in one dimension and $n = 0, 1, \ldots, N-1$ in the other, form a column vector of $x[0]$, $x[1]$, …, $x[N-1]$, and multiply the matrix and vector to find the column vector of $X[0]$, $X[1]$, …, $X[N-1]$.

Appendix B: Code for utplotspec.m

```
function [ftValues, freqs] = utplotspec(x, Ts)
%   UTPLOTSPEC  plot the magnitude and phase of the Fourier
%               transform vs. frequency in Hz and return
%               the frequency-domain samples computed and
%               their corresponding frequency values
%
%   usage  [ftValues, freqs] = utplotspec(x, Ts)
%          ftValues : Fourier transform values
%          freqs : frequencies in Hz corresponding to spectrum vector
%          x : discrete-time signal as a vector of samples
%          Ts : time (in seconds) between adjacent samples in x

% Based on plotspec(x, Ts) from Software Receiver Design
% by R. Johnson, W. Sethares and A. Klein

N = length(x);                     % length of the signal x
freqResolution = 1/(N*Ts);         % frequency resolution fs/N

fx = fft(x);                       % compute DFT/FFT
fxs = fftshift(fx);                % shift it for plotting

if floor(N/2) == N/2               % even
    kstart = -N/2;
    kend = (N/2) - 1;
else                               % odd
    kstart = -(N-1)/2;
    kend = (N-1)/2;
end
kshift = kstart : kend;            % FFT indices for fftshift
freqs = kshift * freqResolution;  % frequency vector

subplot(2,1,1), plot(freqs, abs(fxs));    % plot mag spectrum
xlabel('frequency (Hz)');                 % label axes
ylabel('magnitude (linear units)');
subplot(2,1,2), plot(freqs, unwrap(angle(fxs)));  % plot phase spectrum
xlabel('frequency (Hz)');                 % label axes
ylabel('phase (rad)');

ftValues = fxs;
```