SIO 207A: Fundamentals of Digital Signal Processing
Class 19

Florian Meyer

Scripps Institution of Oceanography
Electrical and Computer Engineering Department
University of California San Diego

0

Short-Time Fourier Transform

- Since many signals are non-stationary random process, a single DFT/FFT over the entire signal would not reveal the finer details of time-varying frequency content
- Thus we are motivated to take DFTs/FFTs over short "windows" of the signal in order to capture the time-varying spectrum
- This leads to the idea of the Short-Time Fourier Transform (STFT)

1

Short-Time Fourier Transform

- The STFT of a signal $x[m]$ is given by

$$X(n,k) = \sum_{m=-\infty}^{\infty} x[n+m]w[m]e^{j\omega_k m} \qquad \omega_k = 2\pi k/N$$
$$= \sum_{m=0}^{N} x[n+m]w[m]e^{j\omega_k m}$$

where $w[m]$ is the analysis window with length N, i.e. , $w[m] = 0$ for $0 > m$ and $m > N$

- As n is increased in the summation, the signal $x[m]$ slides from right to left “through” the analysis window $w[m]$

- For each value of n, the DFT/FFT of $x[n]$, inside the window, is computed

2

Short-Time Fourier Transform

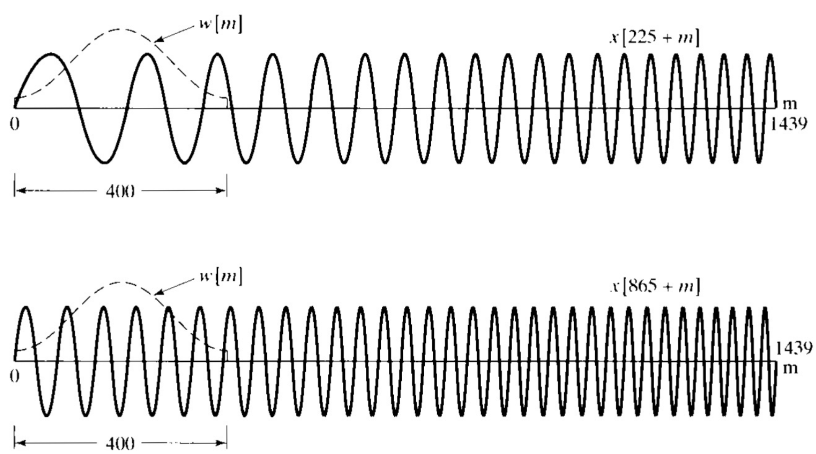
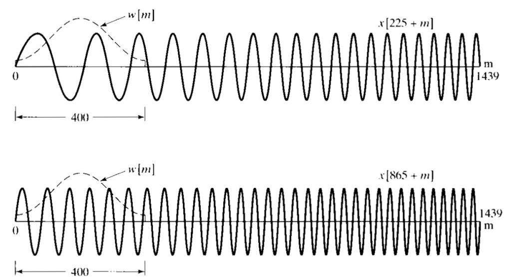


Figure 10.11 Two segments of the linear chirp signal $x[n] = \cos(\omega_0 n^2)$ with the window superimposed. $X[n, \lambda)$ at $n = 225$ is the discrete-time Fourier transform of the top trace multiplied by the window. $X[865, \lambda)$ is the discrete-time Fourier transform of the bottom trace multiplied by the window.

3

Spectrogram

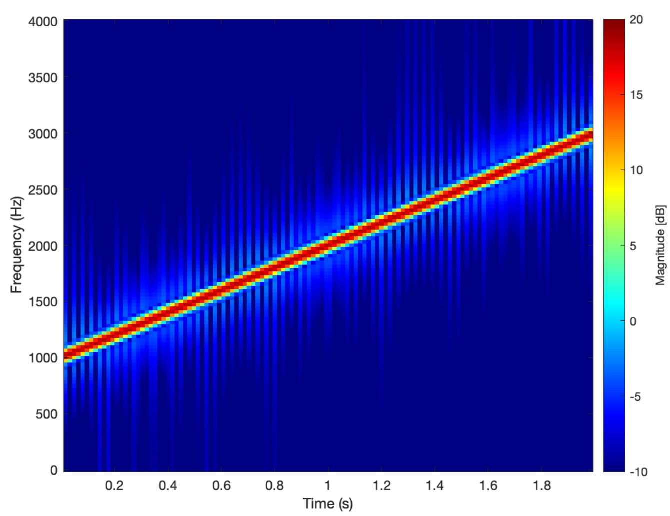
- The collection of DFTs/FFTs (one at each point in time, n) is usually visualized as a spectrogram
- In the spectrogram, we plot the magnitude-squared spectrum, $S(n,k) = |X(n,k)|^2$ vs. n with magnitude values mapped to a color
- Instead of computing the STFT at each time instant, we often slide or advance the window by more than one sample; his is equivalent to computing the STFT every R samples
- Usually the advance is specified in samples, R or as $a\%$ overlap of the window, e.g., 50% overlap



4

Spectrogram – Chirp Example

- Consider a chirp, i.e., a tone whose frequency is linearly increasing



5

MATLAB Code for Spectrogram Visualization

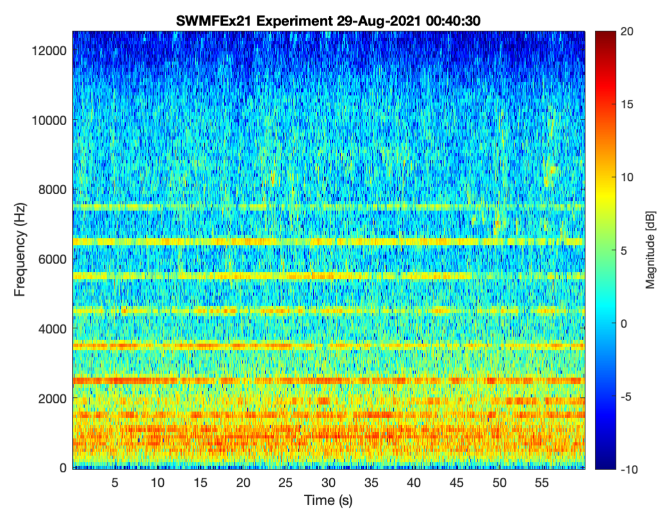
- MATLAB code chirp example:

```
% SIO 207A, Florian Meyer, 2021

clear variables; clc; close all;

% set sampling frequency to 8kHz and length of signal to 2s
fs = 8000;
t = 0:1/fs:2;

% generate chirp between frequencies 1kHz and 3kHz
f1 = 1000;
f2 = 3000;
x = chirp(t,f1,2,f2);

% calculate spectrogram for FFT length of 256 and overlap of 128 samples
nFFT = 256;
overLap = 128;
[S,f,t] = spectrogram(x,hamming(nFFT),overLap,nFFT,fs);

% show spectrogram with dynamic range -10dB - 20dB
imagesc(t,f,20*log10(abs(S)));
set(gca,'ydir','normal'); colormap(jet);
xlabel('Time (s)'); ylabel('Frequency (Hz)');
c = colorbar; c.Label.String = 'Magnitude [dB]';
caxis([-20 35]);
set(gcf,'color','w')
```

6

Spectrogram: SWMFEx21 Underwater Acoustic Data

- 1 minute of data
- Sampling frequency is 25 kHz
- Distance to source is approx. 3 km
- Hydrophone is 130 m deep
- The source transmitted 7 tones at frequencies 1503 Hz, 2503 Hz, 3503 Hz, 4503 Hz, 5503 Hz, 6503 Hz, and 7503 Hz



7

Final Project: Frequency Domain Representation

- Recall that the Fourier transform and inverse transform are given by

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]\, e^{-j\omega n} \qquad \text{periodic with period } 2\pi$$

$$x[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega})\, e^{j\omega n}\, \mathrm{d}\omega$$

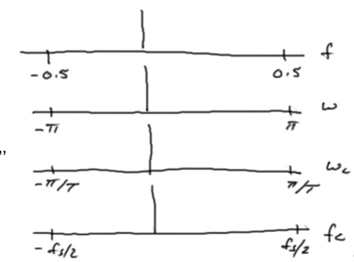
- Four ways to represent frequency domain axis is discrete-time signal processing →

``Normalized Frequency'' f_c/f_s [cycles/sample]

ω_c/ω_s [rad/samples]

``Cont. or Analog Frequency'' [rad/sec]

``Analog Frequency'' [cycles/sec] or [Hz]



(Note that T and $f_s = 1/T$ are sampling interval and sampling frequency, respectively.)

- For final project we always use "normalized frequency" or "analog frequency" in [Hz]

Final Project: Type I and Type II Frequency Domain Rep.

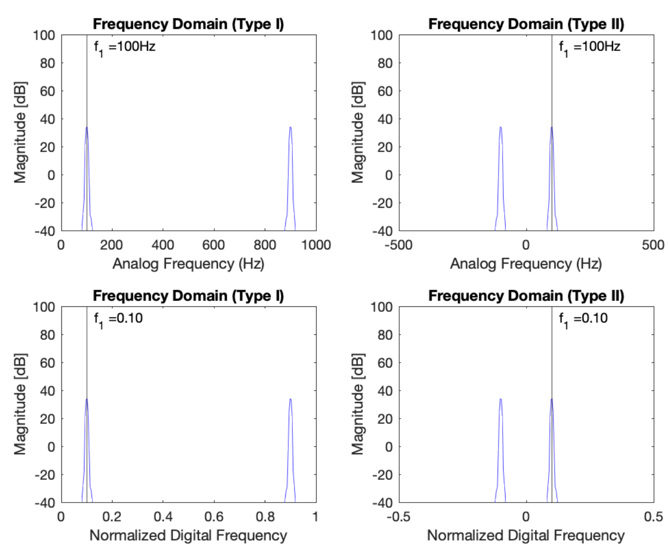
- Matlab Code for Type I (normalized freq.)

```
xFType1 = fft(window.*xTime(1:nFFT));
xFMag1 = abs(xFType1);
freqType1 = (0:1/nFFT:(1-1/nFFT))';
p = plot(freqType1,20*log10(xFMag1));
```

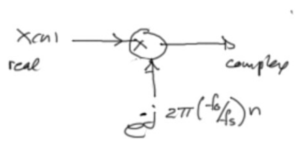
- Matlab Code for Type II (normalized freq.)

```
xFType2 = fftshift(fft(window.*xTime(1:nFFT)));
xFMag2 = abs(xFType2);
freqType2 = (-.5:1/nFFT:(.5-1/nFFT))';
p = plot(freqType2,20*log10(xFMag2));
```

- For final project we always use the Type II representation (we want to make negative frequencies explicit)



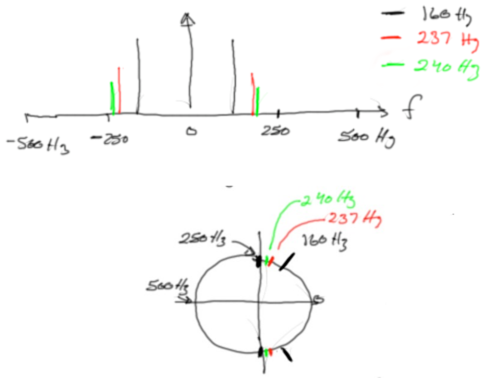
Final Project: Complex Basebanding

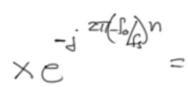


Bandshifting: Recall that multiplication by $e^{-j2\pi(f_0/f_s)n}$ results in a clockwise rotation of the z-transform (see class 5)

$$x[n]e^{-j2\pi f_0/f_s n}$$
$$= x[n]\cos 2\pi f_0/f_s n$$
$$- jx[n]\sin 2\pi f_0/f_s n$$

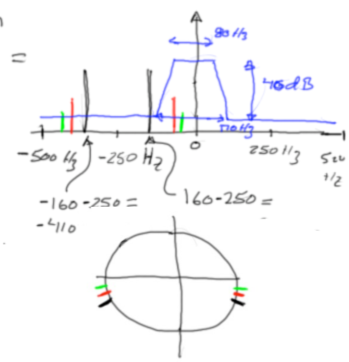
before bandshifting





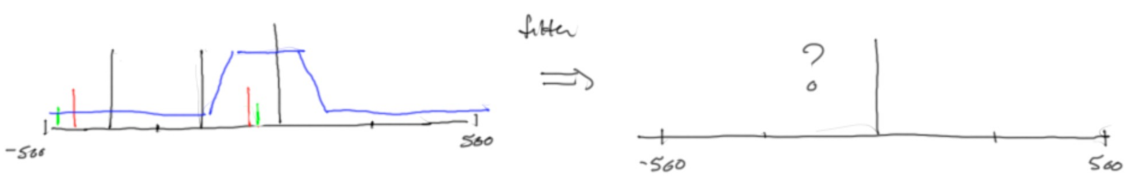
$f_s = 1\text{kHz}$
$f_0 = 250\text{Hz}$

after bandshifting



10

Final Project: Low-Pass Filter

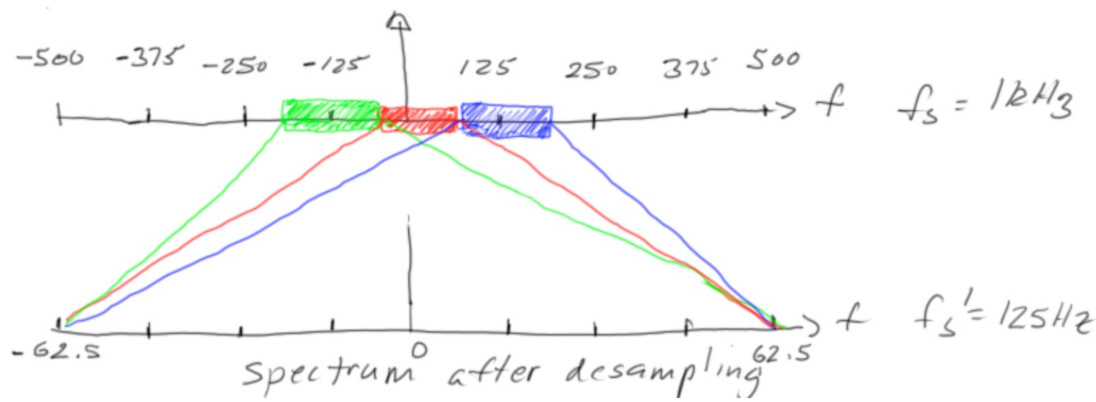


- Subsequently, desample the low-pass filtered complex bandshifted sequence so that $f_s' = f_s/8 = 125\text{Hz}$

11

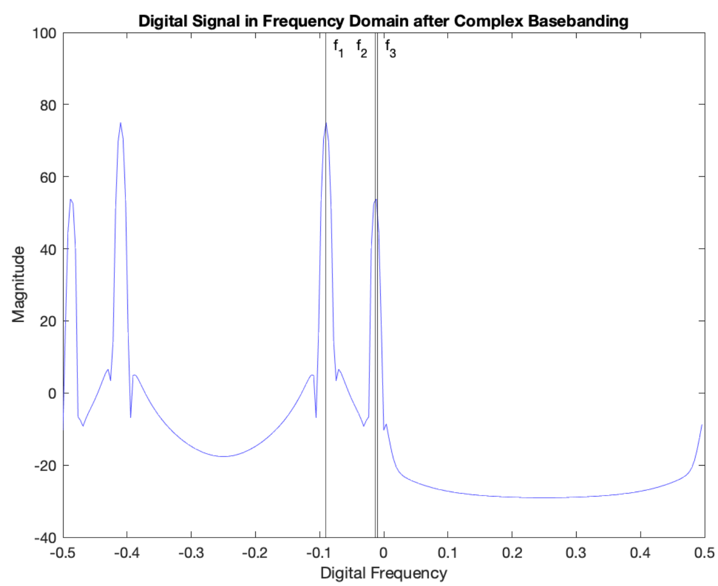
Final Project: Mapping After Desampling

- Spectrum of complex bandshifted signal output of LPF prior to desampling



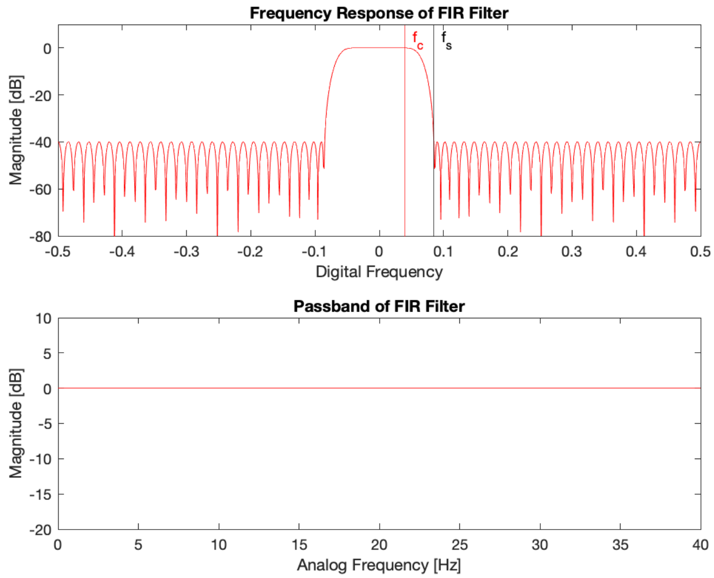
12

Final Project: Result After Basebanding



13

Final Project: Filter Response



14