

% In-Lecture Matlab Assignment #1 on Sept. 14, 2026. Based on homework problem 1.2.

% Key takeaways: (1) Chirp signals are useful in localization, testing and training because they linearly sweep a range of frequencies, and (2) Spectrograms analyze a signal in the time and frequency domains simultaneously so that frequencies can be localized in time. Spectrogram trades off frequency resolution for time resolution.

% **Chirp Signals:** Please see slides 1-14 to 1-16 of [CommonSignalsInMatlab.pptx](#).

% **Spectrograms:** Please see slides 1-17 to 1-20 of [CommonSignalsInMatlab.pptx](#).

% **Introduction:** A chirp signal is a sinusoid whose principal frequency

% increases (or decreases) over time. A chirp signal has the form

% $c(t) = \cos(\theta(t))$ where $\theta(t) = 2\pi(f_0 + 0.5f_{\text{step}}t)t = 2\pi f_0 t + \pi f_{\text{step}} t^2$

% The principal frequency in Hz is f_0 when $t = 0$ and then changes over time at a

% rate of f_{step} in units of Hz/s. The principal frequency of a sinusoid at a given

% point in time is called the *instantaneous frequency*, and it is defined as

% $d\theta(t)/dt$ in units of rad/s. $d\theta(t)/dt = 2\pi f_0 + 2\pi f_{\text{step}}t = 2\pi(f_0 + f_{\text{step}}t)$.

% We divide $d\theta(t)/dt$ by 2π to obtain instantaneous frequency in Hz of $f_0 + f_{\text{step}}t$.

% **(a) Generate a chirp** signal that lasts 10s with $f_0 = 20$ Hz and $f_{\text{step}} = 420$ Hz/s.

% Use sampling rate f_s of 44100 Hz. The chirp will sweep through the principal

% frequencies of the keys on an 88-key piano. Here's Matlab code to get started.

```
%% Generate a chirp signal with frequency increasing
```

```
%% from f0 to (f0 + fstep time) over time seconds
```

```
time = 10;
```

```
f0 = 20;
```

```
fstep = 420;
```

```
fs = 44100;
```

```
Ts = 1 / fs;
```

```
t = 0 : Ts : time;
```

```
%% Add code here to define the chirp signal y = cos( angle(t) )
```

```
angle = 2*pi*f0*t + pi*fstep*t.^2;
```

```
y = cos(angle);
```

% **(b) Play the chirp signal** as an audio signal. Describe what you hear.

% *I hear a rising pitch over time. Sounds like a slide whistle or a tsunami warning siren*

% (rb.gy/18exl). Note: Some laptop playback systems cannot play frequencies below 200 Hz.

```
sound(y, fs);
```

```
pause(time+1);
```

% **(c) Plot the spectrogram** of the chirp signal and describe the visual representation.

% *Spectrogram shows a yellow line that represents the principal frequency in the chirp*

% *signal. The line goes from 20 Hz at time 0s to 4220 Hz at time 10s. The spectrogram*

% *plot is on the next page. See Appendix A for explanation of spectrogram arguments.*

```
figure;
```

```
blockSize = 256; overlap = 128;
```

```
spectrogram(y, hamming(blockSize), overlap, blockSize, fs, 'yaxis');
```

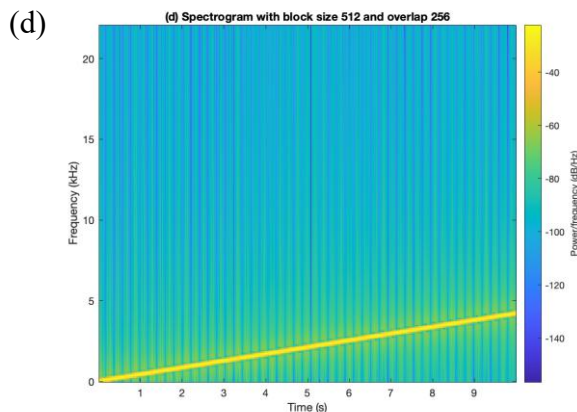
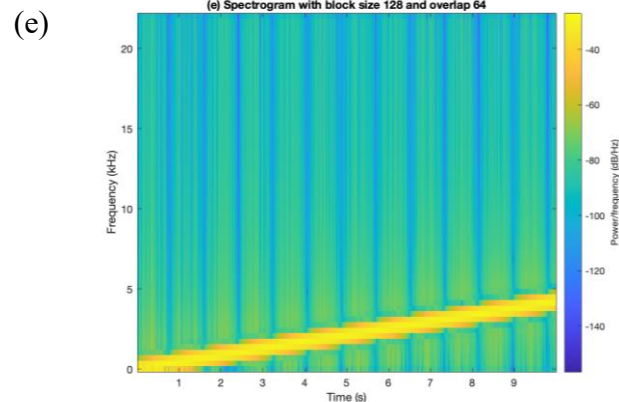
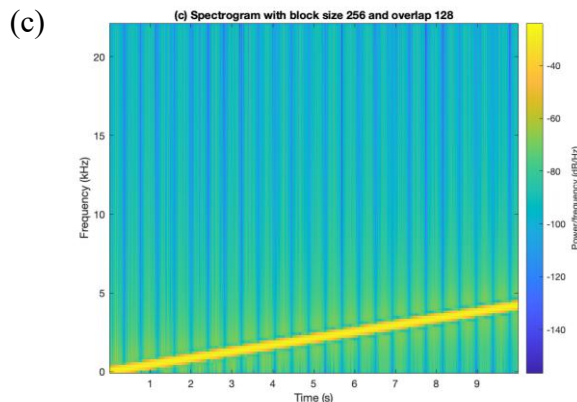
```
title('(c) Spectrogram with block size 256 and overlap 128');
```

(d) Give the code for the spectrogram that would improve the frequency resolution by a factor of two vs. part (c).
*The frequency resolution is what is possible from observing a signal for a block of N samples which lasts for $N T_s$ seconds. From homework problem 0.1, the frequency resolution in Hz is the inverse of the observation time or $1 / (N T_s) = f_s / N$.
 Increase N to decrease (improve) frequency resolution.
 The yellow line in the spectrogram with N doubled is half the width vs. part (c).
 Please see the derivation of frequency resolution in Appendix B.*

```
figure;
blockSize = 2*256; overlap = 128;
spectrogram(y, hamming(blockSize), overlap, blockSize, fs, 'yaxis');
title('(d) Spectrogram with block size 512 and overlap 256');
```

(e) Give the code for the spectrogram that would improve the time resolution, i.e. localizing frequency components in time, by a factor of two vs. part (c).
The time resolution means the ability to identify when a frequency component occurs in time. In a block of N samples, we do not know when frequency components occur, and hence, our time resolution in seconds is $N T_s$. We improve time resolution by reducing N .

```
figure;
blockSize = 256/2; overlap = blockSize/2;
spectrogram(y, hamming(blockSize), overlap, blockSize, fs, 'yaxis');
title('(e) Spectrogram with block size 128 and overlap 64');
```



In all three spectrogram plots, the extent of the horizontal time axis is the same (from 0 to 10s) and the extent of the vertical frequency axis is the same (from 0 to $\frac{1}{2} f_s$ where $f_s = 44100$ Hz). We have chosen f_s to satisfy the sampling theorem $f_s > 2 f_{max}$ where f_{max} is the maximum frequency of interest (4220 Hz) and to be a standard audio sampling rate.

Appendix A: Arguments to the MATLAB spectrogram function by Dan Jacobellis

In HW 1.2 and the in-lecture assignment, a spectrogram is used to visualize the chirp signal.

There are [10 possible input arguments for the spectrogram function in MATLAB](#) [↗] which often leads to confusion.

Here are a few notes about using the spectrogram function in MATLAB.

1. If the output argument is saved, no plot will be generated.

`s = spectrogram(...)` saves the complex-valued DFT coefficients to the variable `s` but does not create a plot.

`figure; spectrogram(...)` creates a new window with the plot of the spectrogram.

2. The `window` parameter has two different uses

If the `window` parameter is an integer, then MATLAB will construct a [Hamming window](#) [↗] of that length, and multiply each frame of data by the hamming window before taking the DFT. This is the suggested mode to use the function, i.e.

`figure; spectrogram(x, 2^10...)`

3. The relationship between time and frequency resolutions is easiest to see when no overlap is used.

Consider the following two spectrograms. Suppose the signal length is $N = 2^{20} = 1048576$

Spectrogram 1:

```
window = 2^10;  
noverlap = 0;  
nfft = 2^10;  
figure; spectrogram(x, window, noverlap, nfft)
```

Spectrogram 2:

```
window = 2^12;  
noverlap = 0;  
nfft = 2^12;  
figure; spectrogram(x, window, noverlap, nfft)
```

The first spectrogram will have $(2^{20} / 2^{10}) = 1024$ divisions on the time axis and $2^{10}/2 = 512$ divisions on the frequency axis (the division by two is because the negative frequencies are discarded). It will result in an image that is 1024 x 512 pixels.

The first spectrogram will have $(2^{20} / 2^{12}) = 256$ divisions on the time axis and $2^{12}/2 = 2048$ divisions on the frequency axis. It will result in an image that is 256 x 2048 pixels.

Both images have the same number of pixels total, but there is a tradeoff in time and frequency resolution.

Appendix B: Derivation of Frequency Resolution

Frequency resolution of Δf Hz means two frequency components spaced Δf Hz apart can each be clearly identified by an algorithm, e.g. well separated in a plot of the frequency domain.

We'll illustrate the concept of frequency resolution by revisiting homework problem 0.1.

Homework 0.1 concerned a sine signal $c(t)$ lasting from 0s to 1s. The mathematical expression is a two-sided sine signal multiplied by a rectangular pulse that lasts from 0s to 1s:

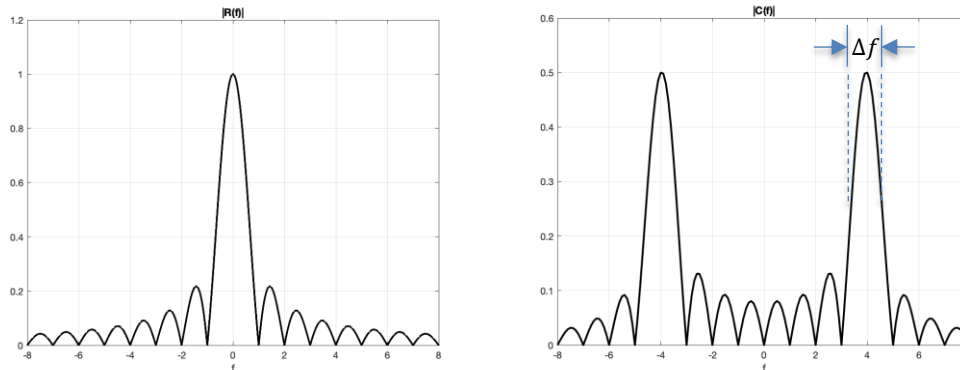
$$c(t) = \sin(2\pi f_c t) \text{rect}(t - 1/2)$$

The continuous-time Fourier transform of $r(t) = \text{rect}(t - 1/2)$ is a sinc function times a phase shift

$$R(f) = F\left\{\text{rect}\left(t - \frac{1}{2}\right)\right\} = \text{sinc}(f) e^{-j\pi f} \text{ where } \text{sinc}(x) = \frac{\sin(\pi x)}{\pi x} \text{ and}$$

$$C(f) = \frac{j}{2} e^{-j\pi(f+f_c)} \text{sinc}(f+f_c) - \frac{j}{2} e^{-j\pi(f-f_c)} \text{sinc}(f-f_c) \text{ due to the modulation property.}$$

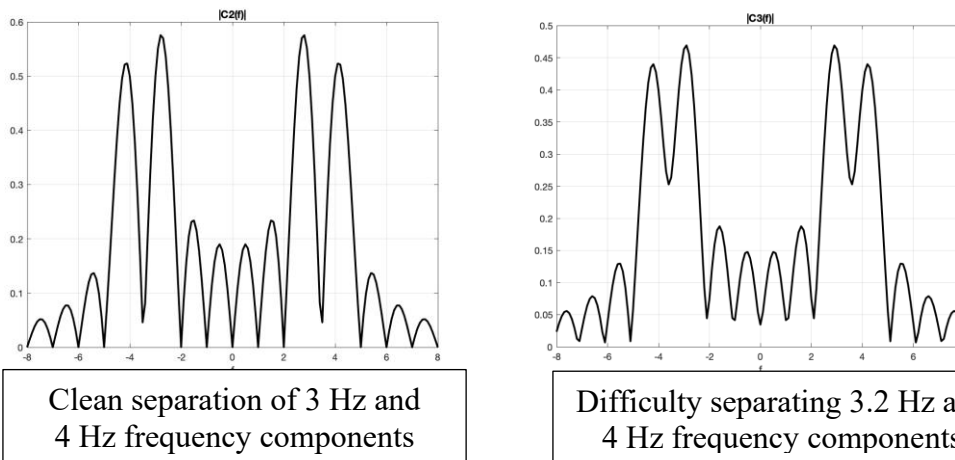
Below are the plots of $|R(f)|$ on the left and $|C(f)|$ for $f_c = 4$ Hz on the right:



For a signal lasting 0s to 1s and containing sinusoids at frequencies 3 Hz and 4 Hz,

$$c_2(t) = \sin(2\pi f_0 t) \text{rect}(t - 1/2) + \sin(2\pi f_1 t) \text{rect}(t - 1/2)$$

let's see if we can resolve the two frequencies. We're looking for two peaks in the frequency domain plot that are well separated at 3 Hz and 4 Hz. Between the peaks, the magnitude response should not be higher than the "sidelobes" at frequencies higher than 1 Hz in $|R(f)|$.



More generally, for a rectangular pulse of duration T seconds, the frequency resolution is $1/T$. The value of $1/T$ is also the null bandwidth.

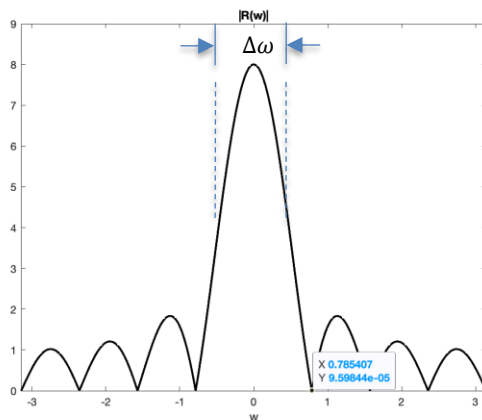
In the course of computing the spectrogram, we apply a rectangular pulse to the discrete-time signal to extract a block of samples to compute their Fourier series coefficients using the fast Fourier transform. Consider a discrete-time signal that is a two-sided sine signal and the first N samples are kept:

$$c[n] = \sin(\omega_c n) \text{rect}((n - N/2)/N)$$

Here, $r[n] = \text{rect}((n - N/2)/N)$ which has amplitude 1 for $n \in \{0, 1, \dots, N - 1\}$ and 0 elsewhere. We can also write $r[n] = u[n] - u[n - N]$ where $u[n]$ is the unit step function. The discrete-time Fourier transform of $r[n]$ is a periodic sinc function times a phase shift :

$$R(\omega) = \underbrace{\frac{\sin\left(\frac{N\omega}{2}\right)}{\sin\left(\frac{\omega}{2}\right)}}_{\text{periodic sinc function}} e^{-j\omega(N-1)/2}$$

where ω is in units of rad/sample. The periodic sinc function is periodic in ω with period 2π . Here's one period of $|R(\omega)|$ for $N = 8$:



```
w = -pi : 0.001 : pi;
N = 8;
Rw = sin(N*w/2) ./ sin(w/2);
figure;
plot(w, abs(Rw), 'k', 'LineWidth', 2);
title(' |R(w)| ');
xlabel(' w ');
ylim([0, 9]);
```

This is the magnitude response of an averaging filter with 8 coefficients. Please see the [Designing Averaging Filters](#) [handout](#).

The first zero for the magnitude response in positive frequencies occurs at $2\pi / N$. This is the null bandwidth and also the frequency resolution $\Delta\omega$.

Let's connect the frequency resolution in the discrete-time frequency domain to the continuous-time frequency domain:

$$\Delta\omega = \frac{2\pi}{N} = 2\pi \frac{\Delta f}{f_s} \text{ means that } \Delta f = \frac{f_s}{N}$$