% In-Lecture Assignment #1 on Wednesday, Sept. 15, 2021

% Based on homework problem 1.2**.** Dan's comments are on the last page.

% Key takeaways: (1) Chirp signals are useful in localization, testing and training
% because they linearly sweep a range of frequencies.
% (2) Spectrograms analyze a signal in the time and frequency domains
% simultaneously so that frequencies can be localized in time. The
% 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
% $c(t) = \cos(\ q(t)\ )$ where $q(t) = 2 \pi\ (\ f_0 + 0.5\ f_{step}\ t\ )\ t = 2 \pi\ f_0\ t + \pi\ f_{step}\ t^2$
% The principal frequency 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
% $dq(t) / dt$ in units of rad/s. $dq(t) / dt = 2\ p\ f_0 + 2\ p\ f_{step}\ t = 2\ p\ (f_0 + f_{step}\ t)$.

% **(a) Generate a chirp** signal that lasts 10s with f_0 = 20 Hz and f_{step} = 420 Hz/s.
% Use sampling rate f_s of 44100 Hz. The chirp will sweep through the frequencies
% of the keys on an 88-key piano.
% Here is Matlab code to help you 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 + 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.*
% *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 using the spectrogram

% function in Matlab and describe the visual representation.

% Spectrogram shows a line that represents the principal frequency in the
% chip signal. The line goes from 20 Hz at time 0s to 4220 Hz at time 10s.
% The spectrogram plot is on the next page.

```
figure;
blockSize = 256;
overlap = 128;
spectrogram(y, hamming(blockSize), overlap, blockSize, fs, 'yaxis');
```

% **(d) Give the code** for the spectrogram that would improve the
% fequency resolution by a factor of two.

% Frequency resolution is fs / N. Increase N to get better frequency resolution.

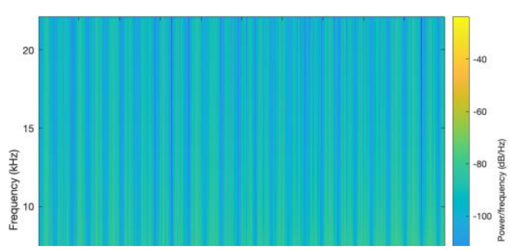
```
figure;
blockSize = 2*256;
overlap = 128;
spectrogram(y, hamming(blockSize), overlap, blockSize, fs, 'yaxis');
```

% **(e) Give the code** for the spectrogram that would improve
% the time resolution by a factor of two.

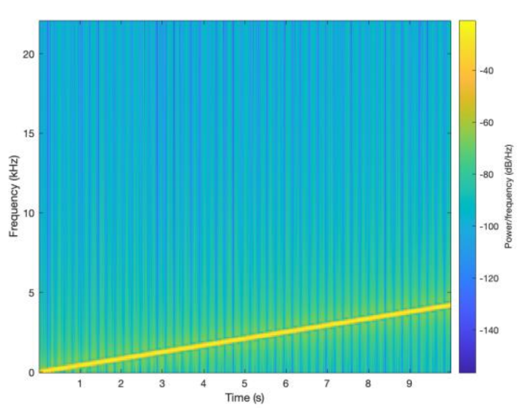
% Time resolution is N. Decrease N to get better time resolution.
% Note: By changing blockSize to 128, it's the same value as the overlap parameter,
% which means that there's no shift and Matlab will generate an error. We'll also
% reduce the overlap so that there's a shift.

```
figure;
blockSize = 256/2;
overlap = 128/2;
spectrogram(y, hamming(blockSize), overlap, blockSize, fs, 'yaxis');
```

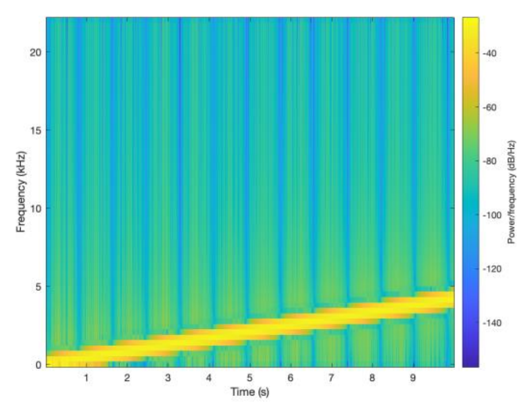
(c)



(d)



(e)



In-lecture assignment deadline is 11:59pm. Notes on MATLAB Spectrogram function. Alternative methods for time-frequency analysis.

Dan Jacobellis (He/Him/His)

All Sections

Sep 15 at 3:18pm

Hi everyone,

I've received a few questions regarding the MATLAB spectrogram function used on the in-lecture assignment, so we've extended the deadline to 11:59pm so that you have an opportunity to discuss it during office hours if necessary.

Parameters used in the MATLAB spectrogram function

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.

I encourage you to try different parameters and see how it affects the spectrogram.