

# Basic Audio Signal Analysis

Fundamentals of Acoustics and Sound  
Topic 1

Christian Sejer Pedersen



AALBORG UNIVERSITY  
DENMARK

# Agenda

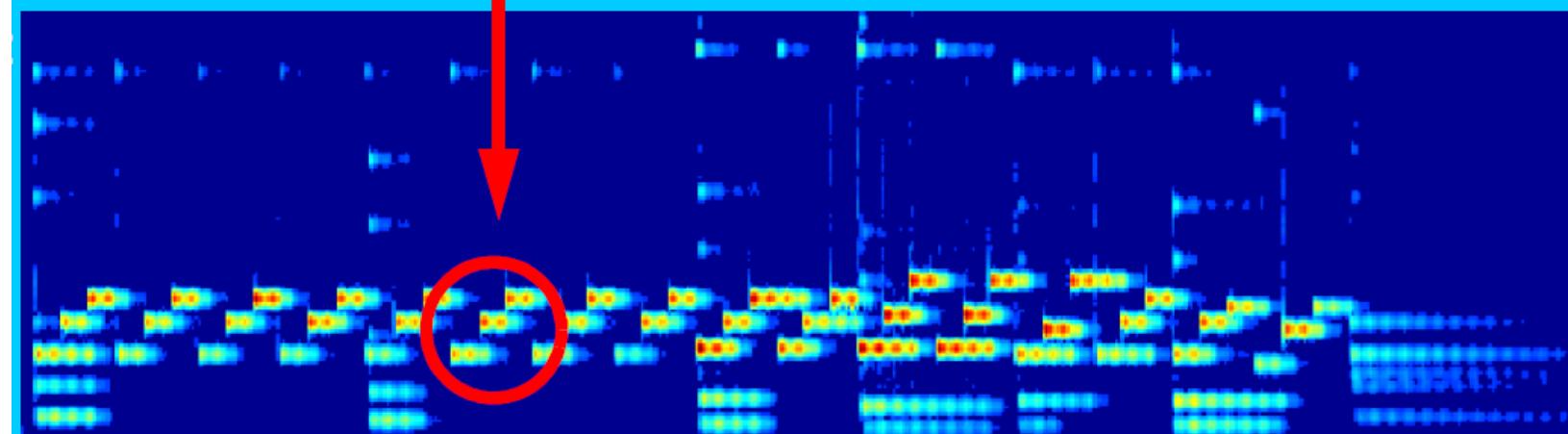
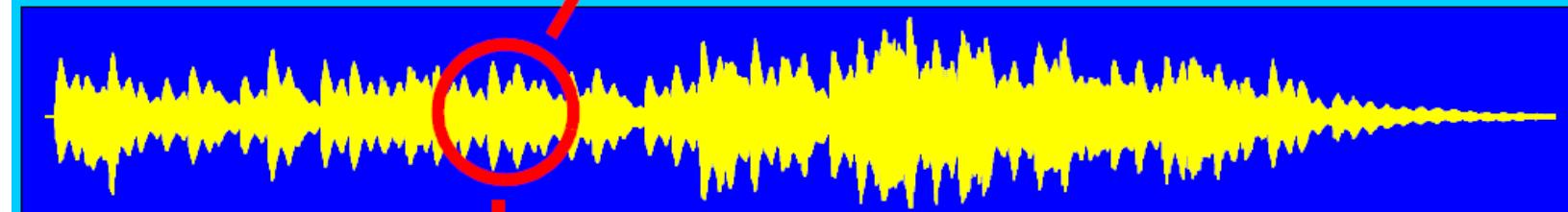
- Intro to course
- Basic Audio Signal Analysis
- Sound as digital audio signals
- Fast Fourier Transform (FFT)  
and the frequency domain
- Time Windows
- Short Time Fourier Transform (STFT)
- Spectrograms
- Exercises

## SONATE

Sonata quasi una Fantasia  
Op. 27, Nr. 2

### Adagio sostenuto

Si deve suonare tutto questo pezzo delicatissimamente e senza sordino.



# Intro to course

The course covers many different aspects of acoustics and sound. Examples of AI applications will be given where applicable.

**Matlab** will be the tool used for many lectures.

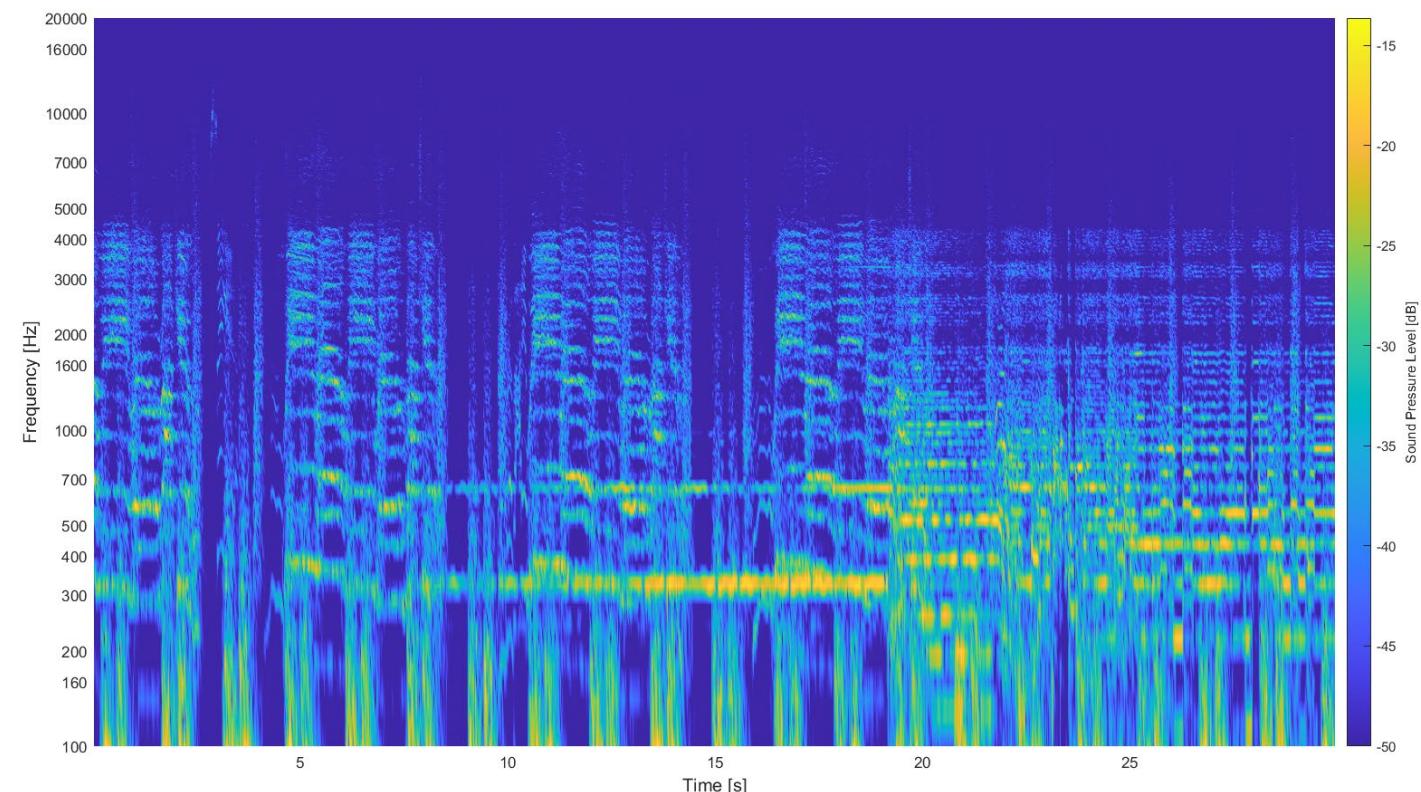
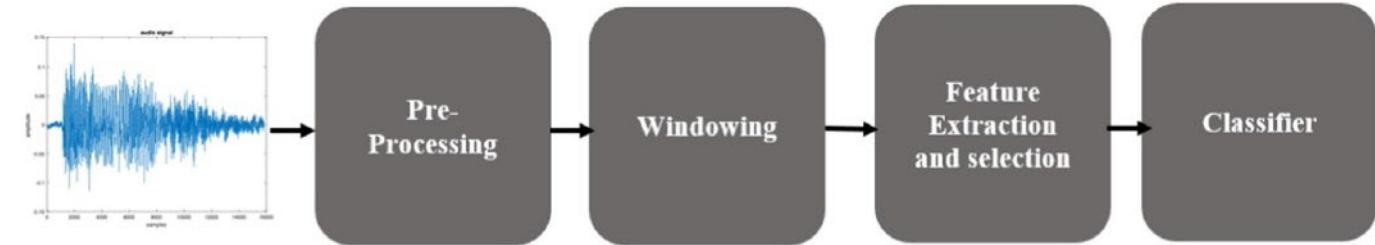
The course will assist in understanding and getting the best audio signal data for your project  
(CE8-AVS students)

The exam will be a written exam based on multiple choice questions

1	CP	<b>Basic Audio Signal Analysis</b> Digital signals, FFT, Windowing, STFT, Spectrograms
2	CP	<b>Feature extraction of audio signals</b> Common features for AI in both time domain, frequency domain and time-frequency domain
3	DH	<b>Basic physics of sound</b> Basic definitions, Wave equation, Harmonic plane wave, Media shifts (Welcome to simulations)
4	CP	<b>System response measurements</b> System response, Impulse response, sweeps, Laboratory hands-on, (Welcome to lab)
5	CP	<b>Sound localization and Beamforming</b> Simple localization with two microphones (cross-correlation), beam forming (Cover both acoustic cameras, hearing aids, Acoustic camera hands on)
6	CP	<b>Rooms</b> Rectangular rooms , Absorption, Reverberation, Architectural acoustics, Room simulation methods
7	FC	<b>Musical Instruments</b> Sound generation, principles, Acoustic fingerprint, Recording, studio , Synthesizers, Sound effects
8	CP	<b>Loudspeakers</b> Transduction, The moving coil loudspeaker, Boxed speakers, Multi-way systems
9	CP	<b>Stereo and beyond</b> Loudspeaker-based VR (SDM), Sound field control (eq, sound zones loudspeaker beamforming),
10	DH	<b>Headphones, (w)earables</b> Headphone technology, Transfer functions and desired targets, Earables and options for automation

# Why is this lecture important?

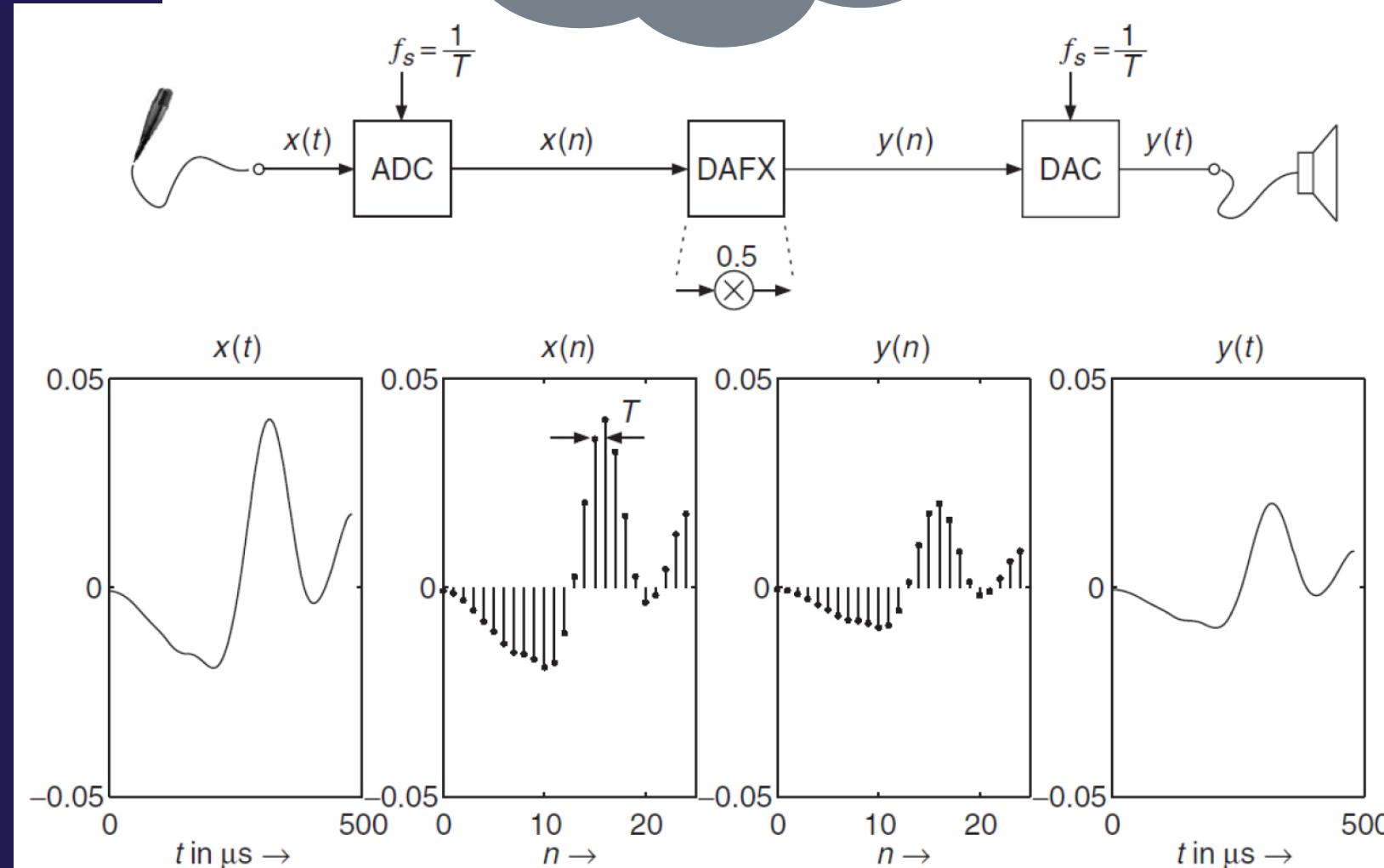
- ⦿ Audio signal analysis is the foundation for any type of AI on audio signals.
- ⦿ This lecture leads up to next lecture where we will explore typical audio features used for AI.
- ⦿ A spectrogram is a very powerful method for visualizing and interpreting audio signals and therefore a very used basis for many AI methods. Can turn audio signal into an image.  
A spectrogram may reduce the size of the data used e.g. 16 bit 44.1 kHz 20 second wave file (1.76 Mb) into an image of e.g. 1024x256x3 (3x8 bit rgb color image=0.77 Mb uncompressed bitmap).  
However, it is important that the spectrogram captures the important information about the audio signal.
- ⦿ Understanding the fundamentals and pitfalls is important to ensure that the AI can derive the important information in the signal.
- ⦿ Therefore, todays lecture focus on the fundamentals and getting hands on experience with the frequency and time-frequency analysis of audio signals.
- ⦿ Also, I will introduce you to Matlab, which will be used in many lectures in the course.



# Sound as a digital audio signal

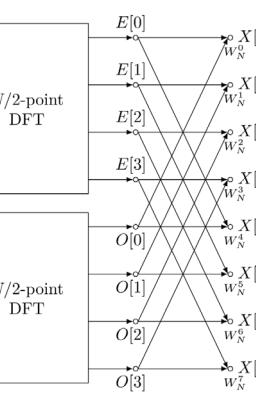
- Sound is tiny pressure fluctuations that can be transformed into voltage using a microphone.
- The voltage can be amplified and sampled by an analog-to-digital converter at a specific sampling frequency  $f_s > 2f_{\max}$  so typically:  
44.1 kHz or 48 kHz  
or integer multiples of these
- The resolution in amplitude is in *bits*, typically 16 or 24-bit:  
 $16^2 = 65536$   
 $24^2 = 16777216$
- In Matlab this is converted into double precision floating point values between -1 and 1.

We can analyze and/or process the audio signal digitally and eventually convert it into sound again!



# Fast Fourier Transform (FFT)

- Fourier Transform convert a signal from time domain into the frequency domain (and back again to time domain using the Inverse Fourier Transform)
- In practice we have a discrete digital signal ( $1/f_s$  timestep for each sample) and a limited time window ( $N$  samples – also called *Blocksize*) → Discrete Fourier Transform
- The fastest way to calculate this exploits symmetry and is called:  
Fast Fourier Transform (FFT) and is fastest when  $N$  is a power of 2.  
<http://www.fftw.org/speed/Ryzen-7-3.6GHz/>



Fourier transform

Inverse Fourier transform

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-j2\pi ft} dt$$

$$\omega = 2\pi f$$

$$x(t) = \int_{-\infty}^{\infty} X(f) e^{j2\pi ft} df$$

DFT

IDFT

Notice that  $N$  is the number of “frequency steps” → “frequency resolution”

$$X_k = \sum_{n=0}^{N-1} x_n e^{-j2\pi \frac{k}{N} n}$$

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{j2\pi \frac{k}{N} n}$$

Notice infinity!

Notice complex number!

# FFT spectrum example with Matlab code

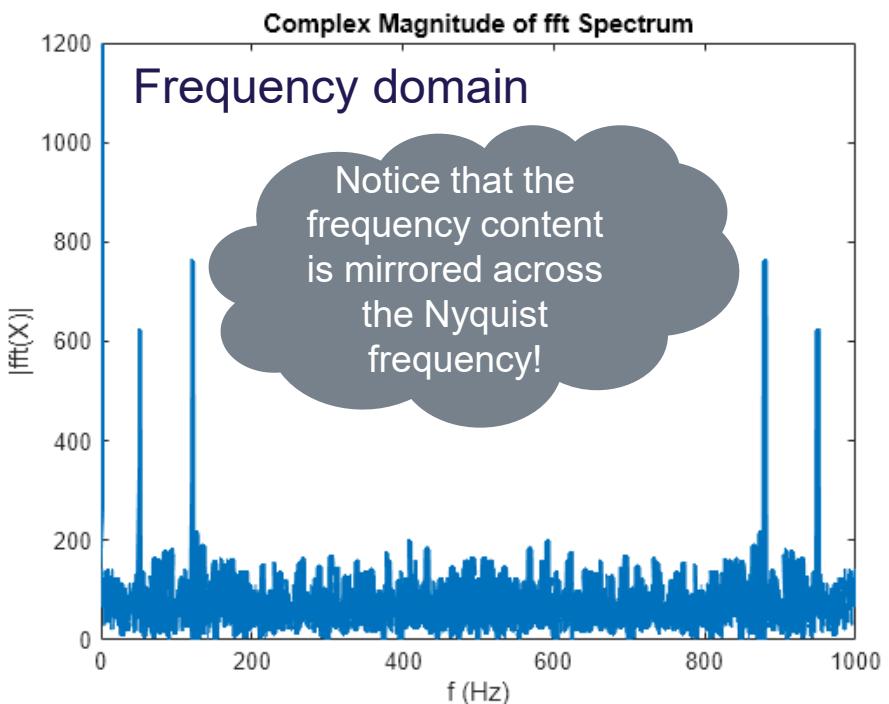
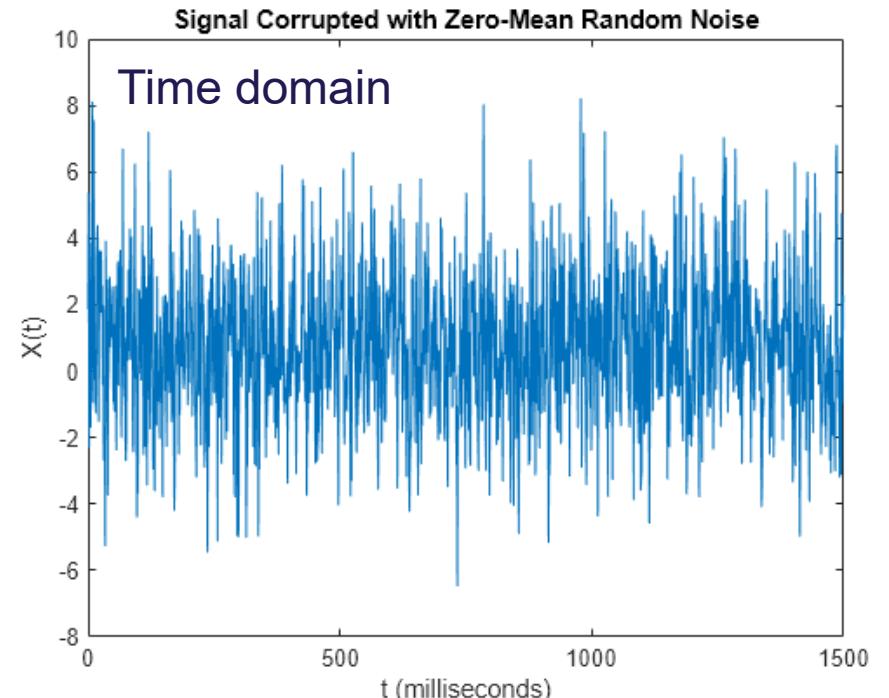
```
① Fs = 1000; % Sampling frequency
   T = 1/Fs; % Sampling period
   L = 1500; % Length of signal
   t = (0:L-1)*T; % Time vector
   S = 0.8 + 0.7*sin(2*pi*50*t) + sin(2*pi*120*t); %two sinuoids
   X = S + 2*randn(size(t)); %Noise added to signal

② plot(1000*t,X)
   title("Signal Corrupted with Zero-Mean Random Noise")
   xlabel("t (milliseconds)")
   ylabel("X(t)")

③ Y = fft(X); %Fast Fourier Transform

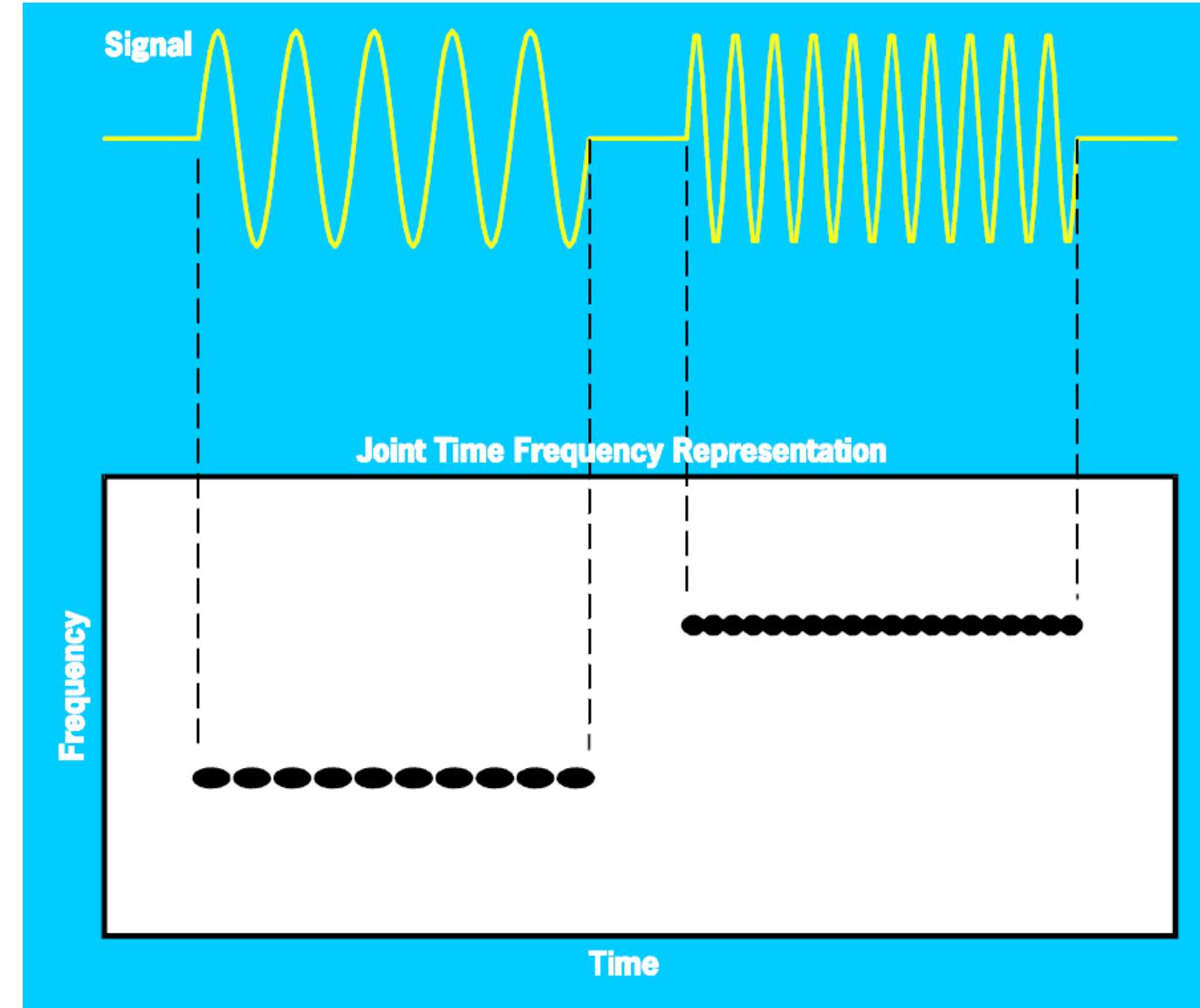
④ plot(Fs/L*(0:L-1),abs(Y),"LineWidth",3)
   title("Complex Magnitude of fft Spectrum")
   xlabel("f (Hz)")
   ylabel("|fft(X)|")
```

fft() will by default  
use the number of  
samples in the time  
signal as N



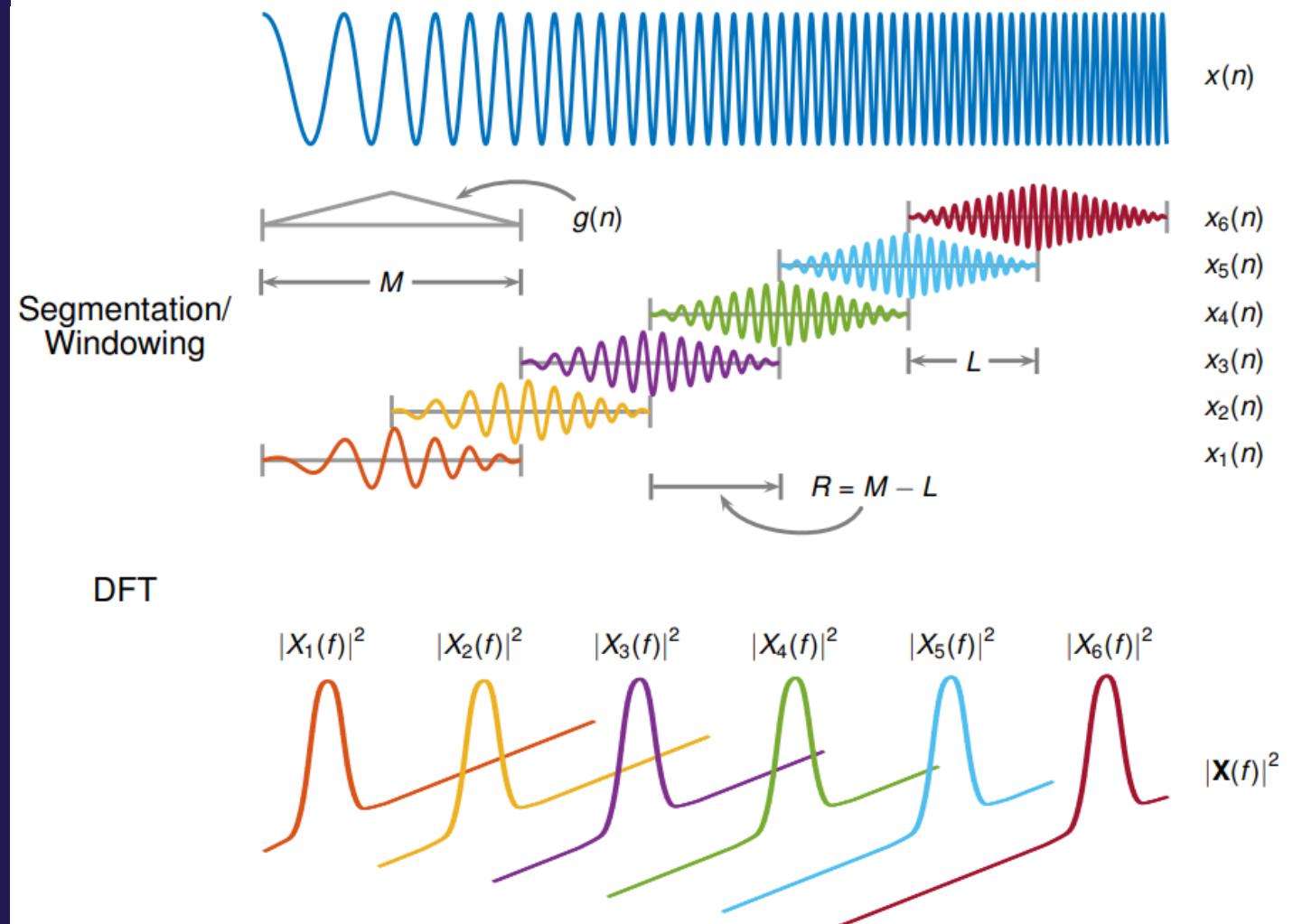
# The FFT gives us the frequency Spectrum

- ➊ The FFT is very useful for analyzing a stationary signal.
- ➋ But what happens if the signal changes over time?
- ➌ The frequency spectrum only shows the average frequency content for the whole time window!
- ➍ **But we would like to see the change in audio over time?**  
Any suggestion to how to obtain that?



# Short Time Fourier Transform (STFT)

- ⦿ Solution! → make multiple smaller time windows and take FFT of each → STFT.
- ⦿ This will provide the frequency content at each window, however, we need to understand the uncertainty principle in order to understand the limitations of this approach.



# Uncertainty principle

Windowing introduction of “smear” in the resolution

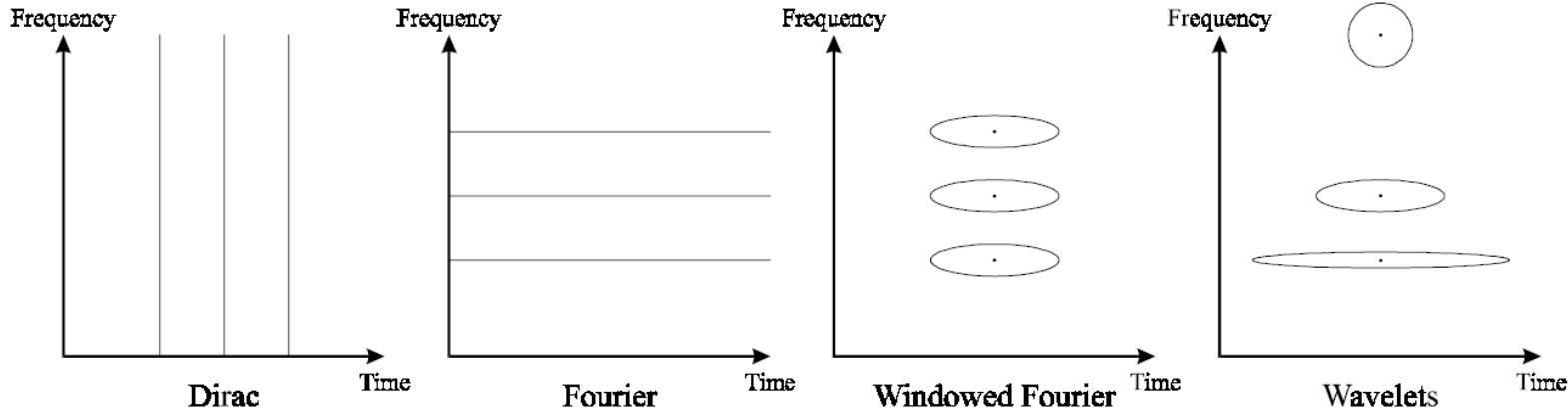
Total amount of information

$$\Delta f \approx \frac{1}{T}$$

$$\frac{F}{\Delta f} \approx F \cdot T$$

Frequency domain

Complete description of the system is given with  $F \cdot T$  numbers



$$\Delta t \approx \frac{1}{F}$$

$$\frac{T}{\Delta t} \approx F \cdot T$$

Time domain

There is a minimum size of the Heisenberg ellipse!!

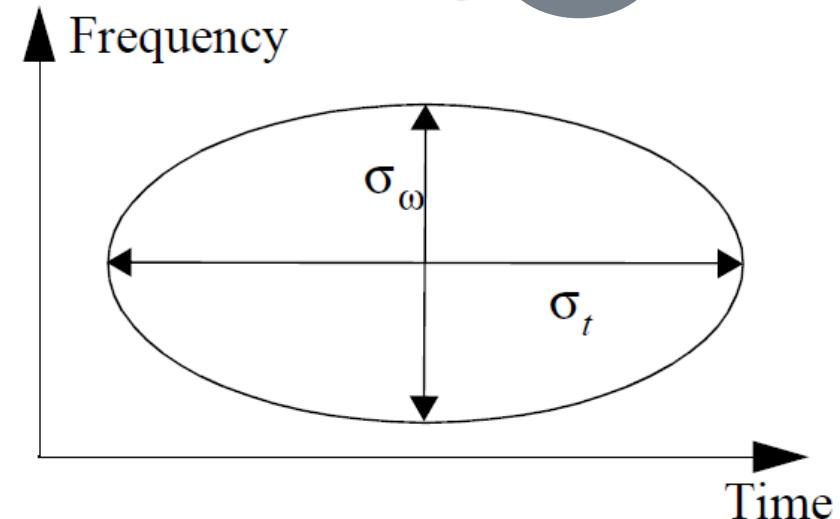


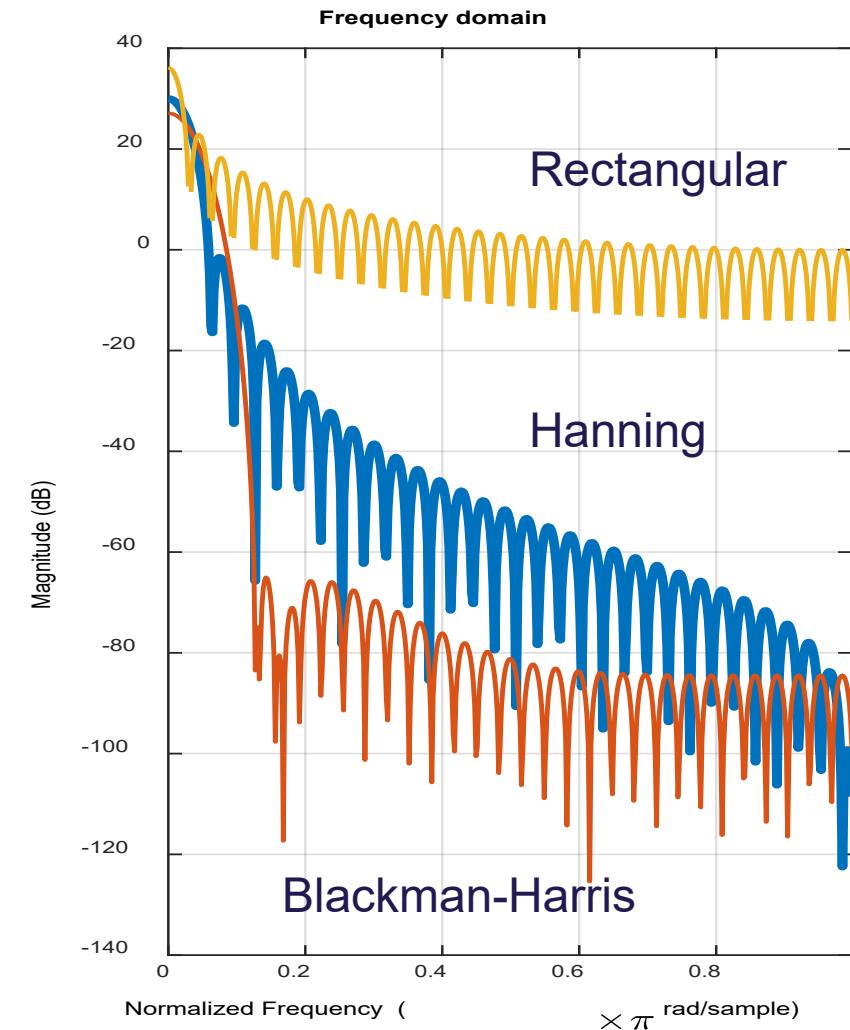
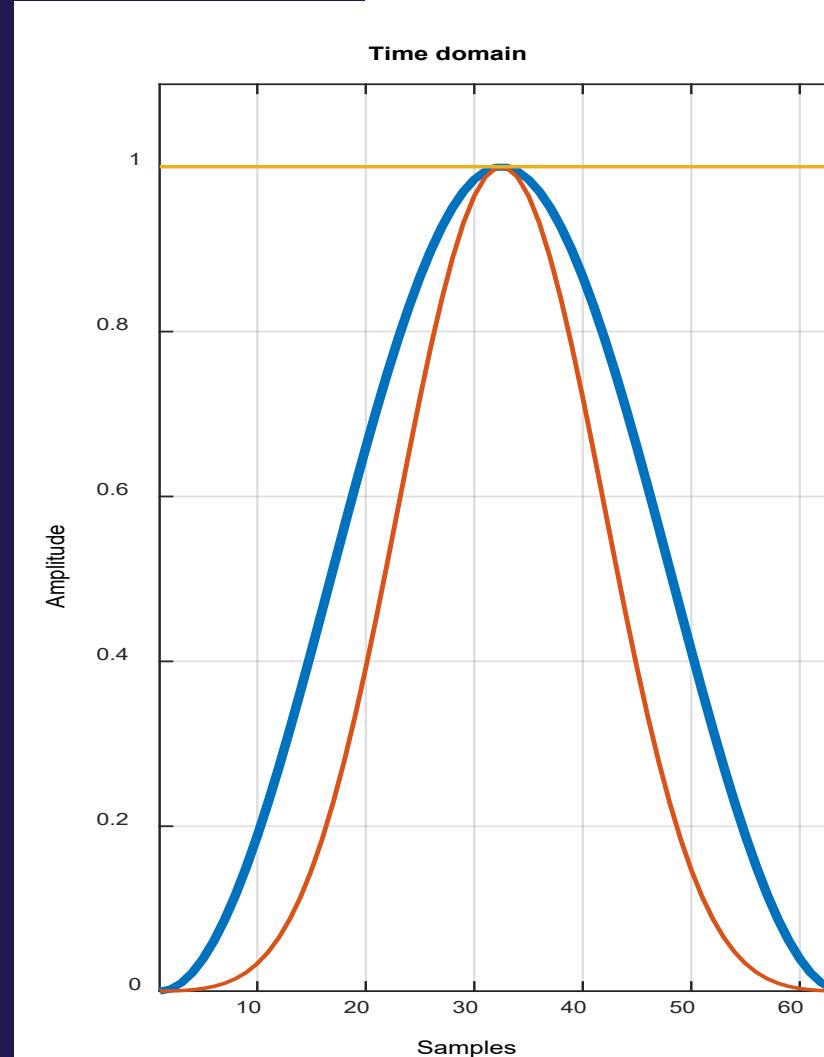
Figure 2.1 Heisenberg ellipse

# Spectral leakage and windows

- ▶ A typical time window tapers off at beginning and end → reduces spectral leakage caused by irregularities in signal at beginning/end of the window 😊
- ▶ However, there is no free lunch 😞 – the main lobe becomes wider i.e. the spectral resolution becomes lower → it is more difficult to separate two nearby tonal components.

These are just examples – many more windows exist!

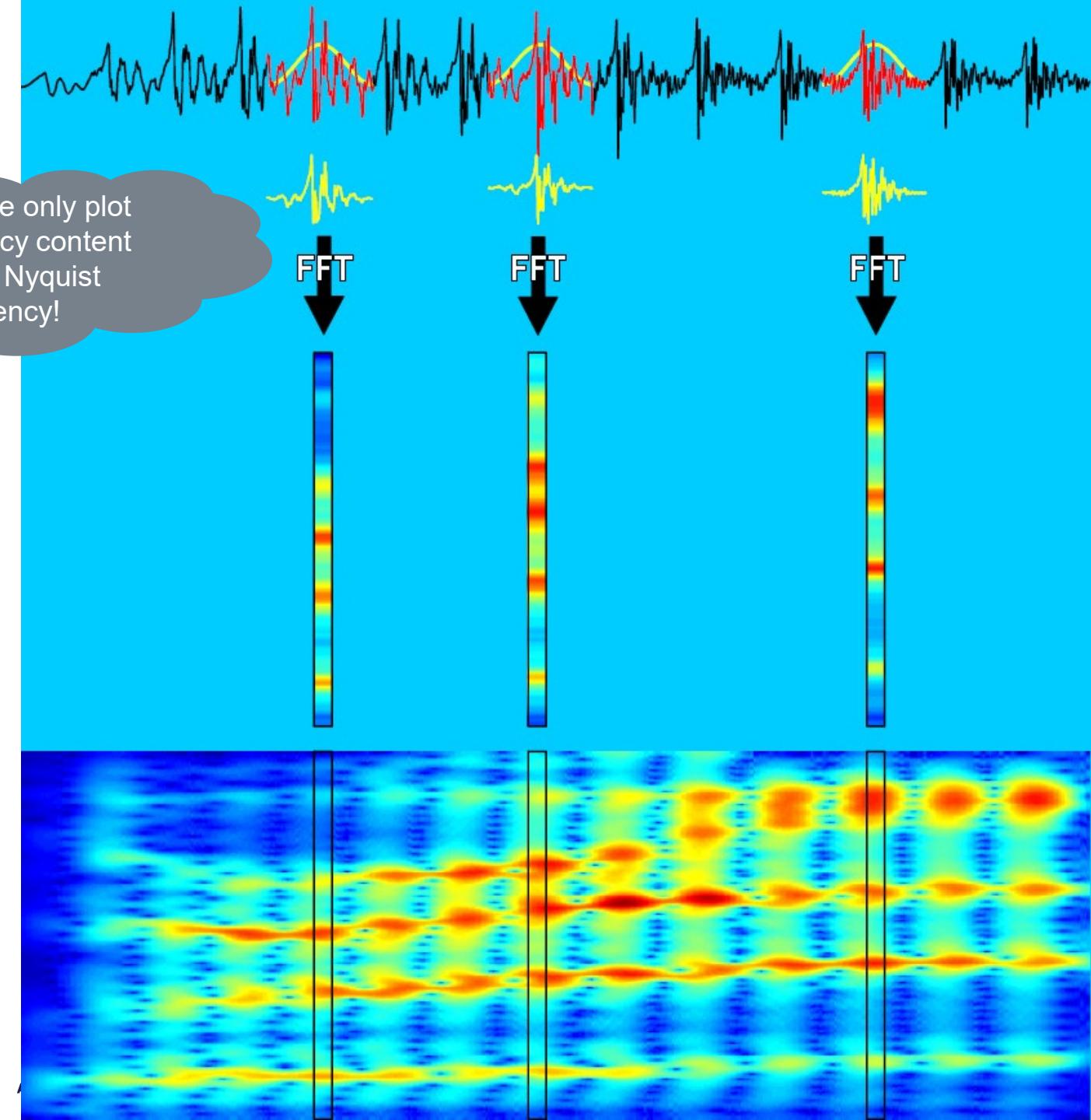
Use `windowDesigner` in Matlab to explore



# Spectrogram using STFT

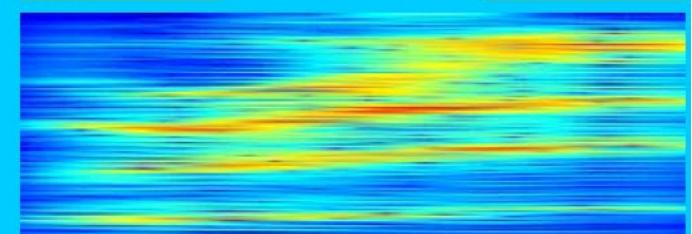
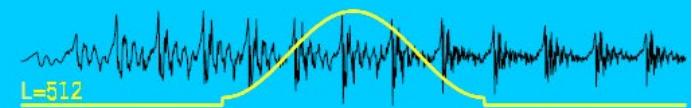
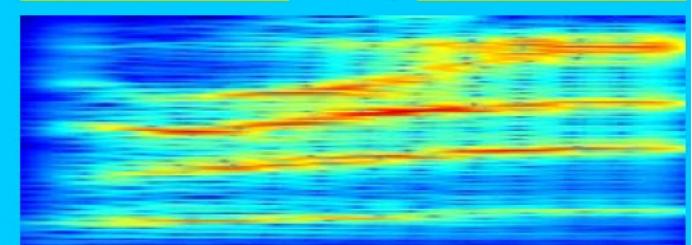
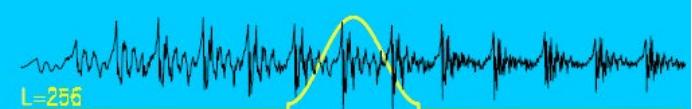
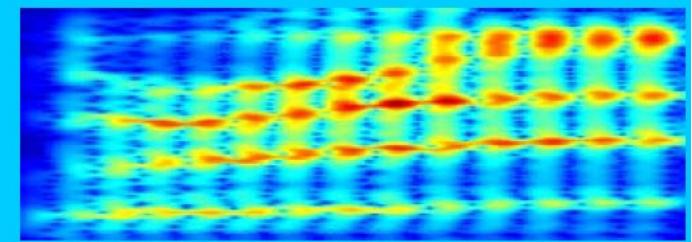
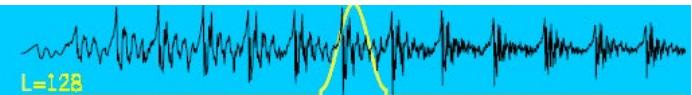
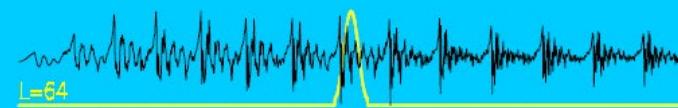
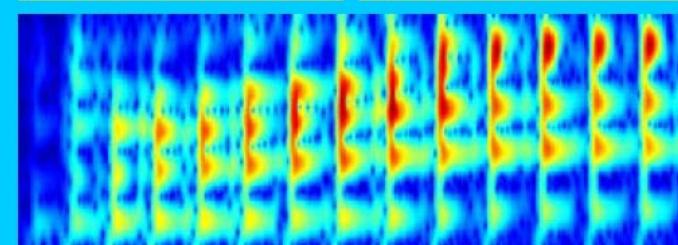
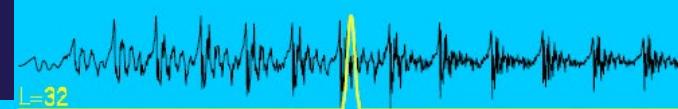
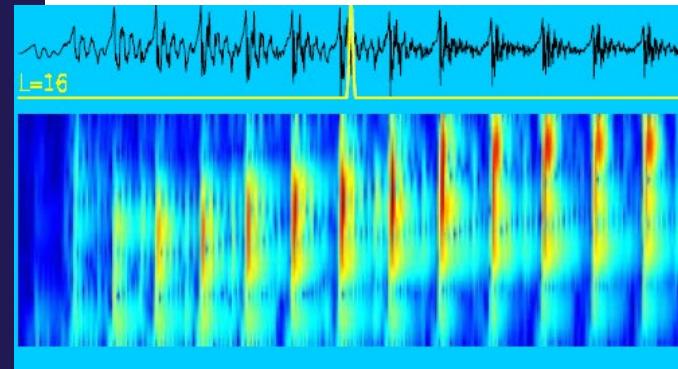
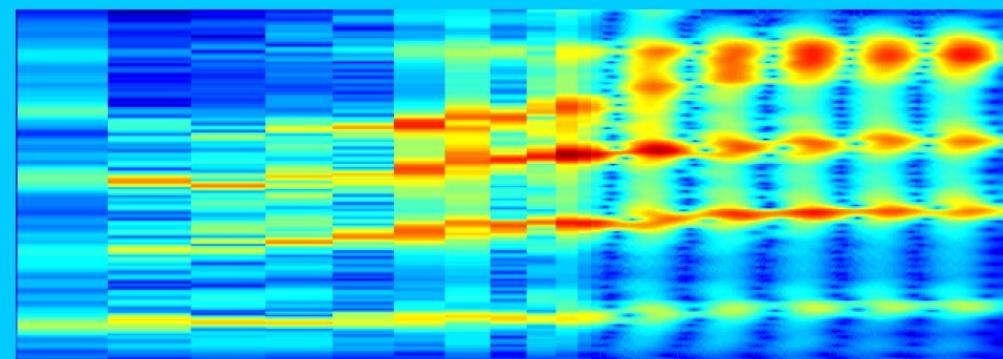
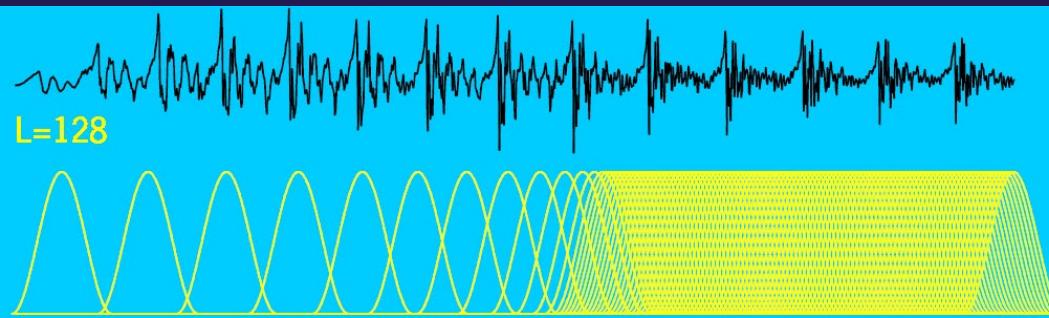
Typically, we only plot the frequency content up to the Nyquist frequency!

- ➊ A spectrogram can be constructed by plotting the absolute amplitude (or power) for each time window FFT and color it according to the magnitude.
- ➋ Time is typically on the x-axis and frequency on the y-axis
- ➌ The spectrogram information depend very much on your choice of window parameters (i.e. type, size and % overlap).



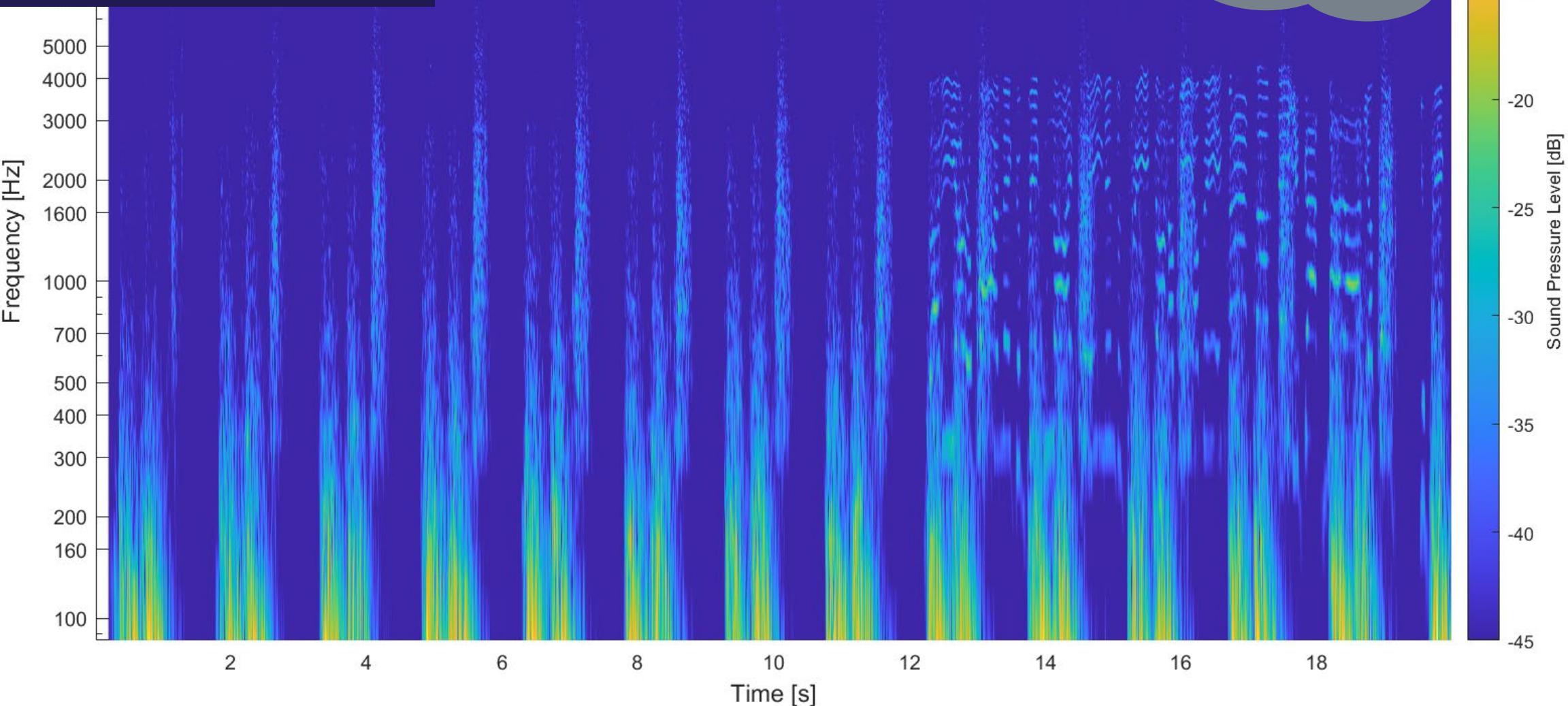
# Spectrogram and effect of window

- The length and shape of the time window and the % overlap are important regarding spectral leakage and having correct amplitude information and simple reconstruction afterwards – for this 50% is often the best choice.



When does the vocal start?

# Spectrogram Interpretation

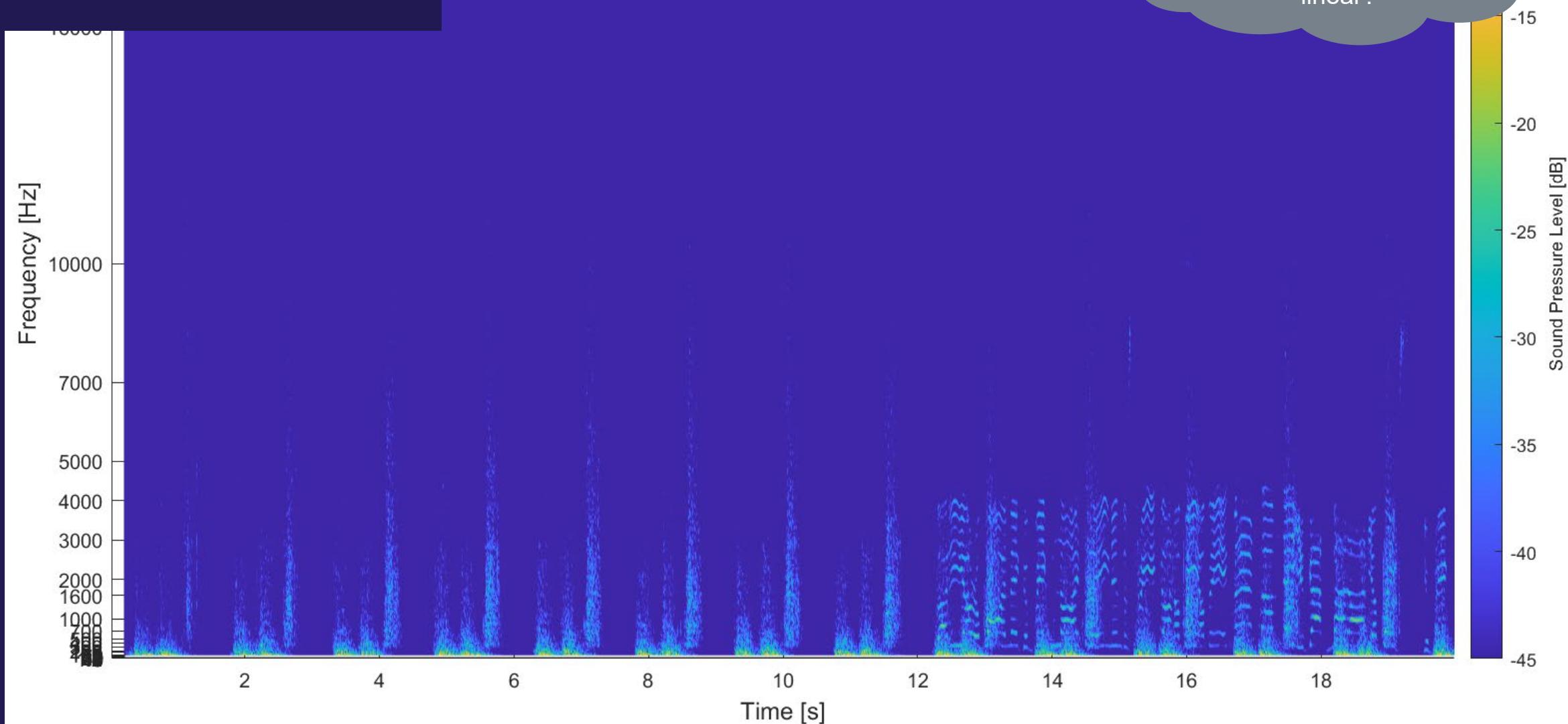


Notice that I have plotted with log frequency scale  
Why did I do that?

# Spectrogram Interpretation

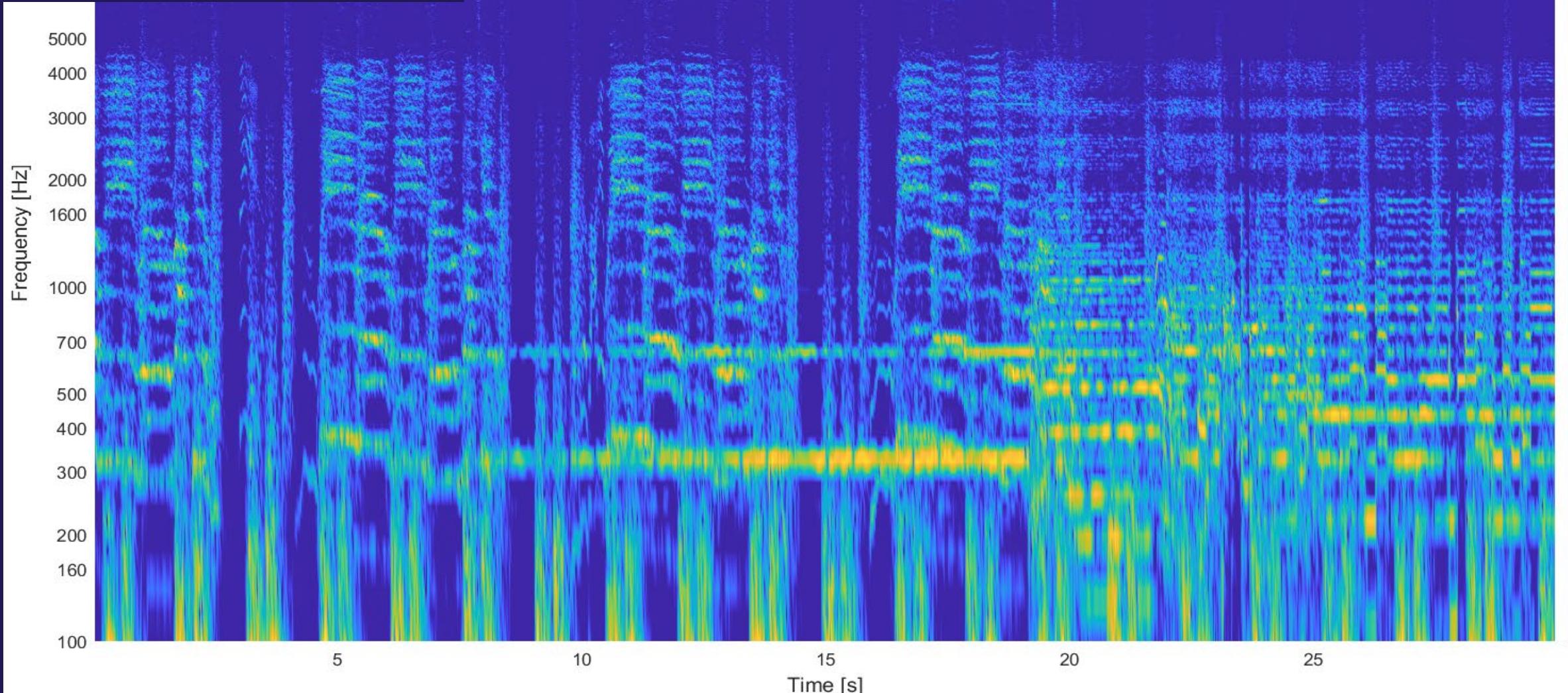


## When does the vocal start?



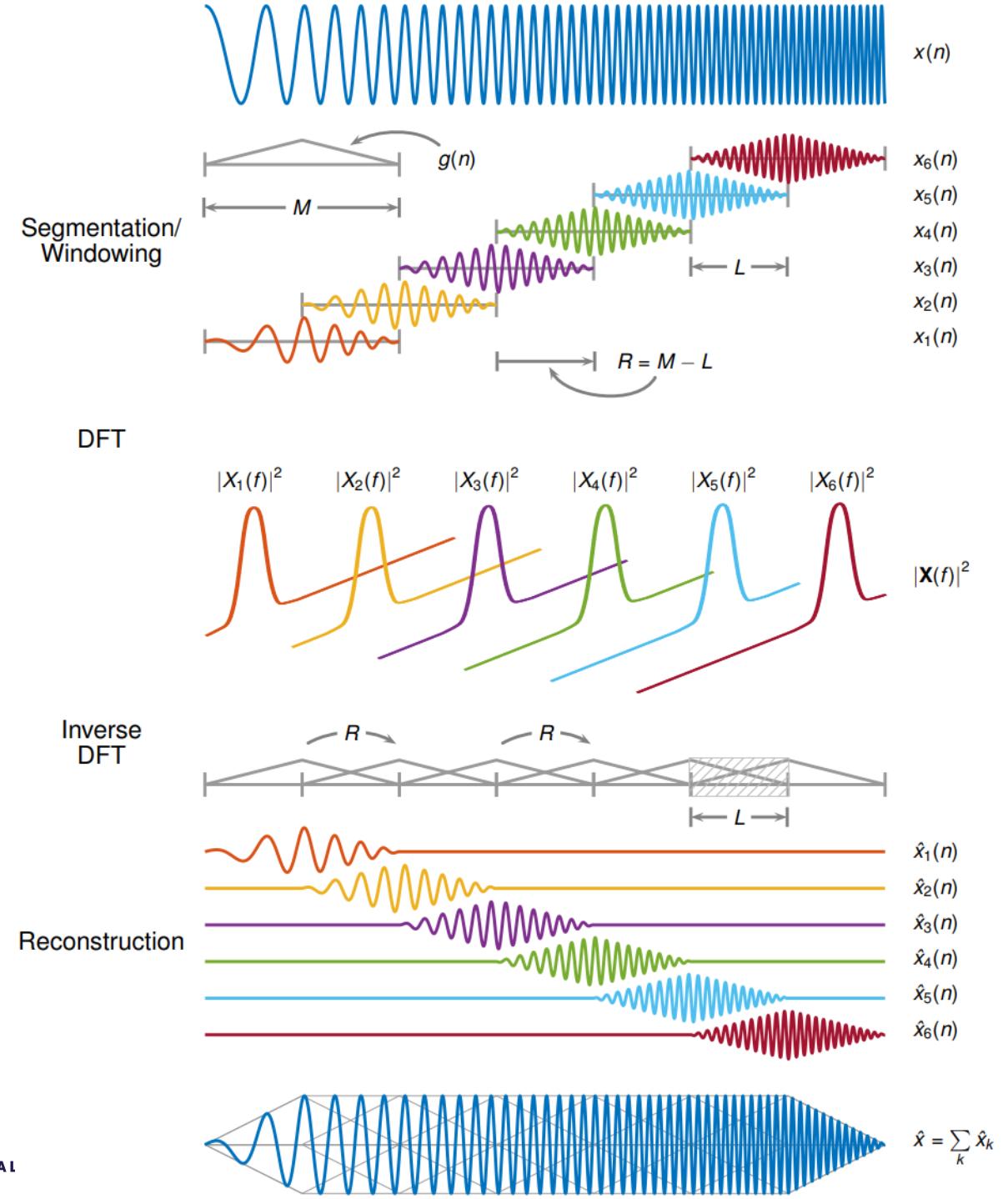
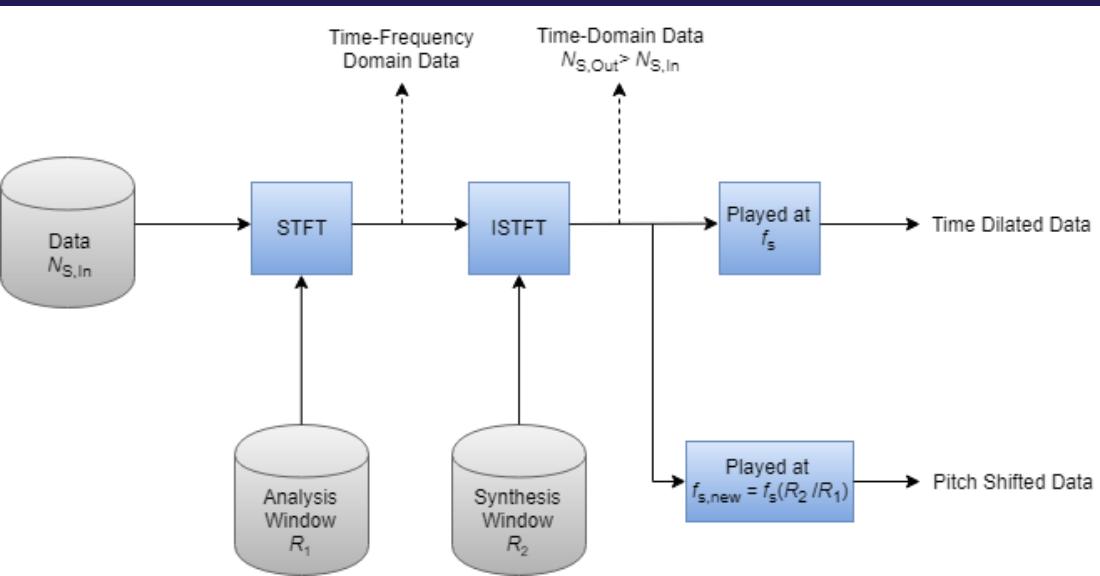
When does the guitar start?

# Spectrogram Interpretation



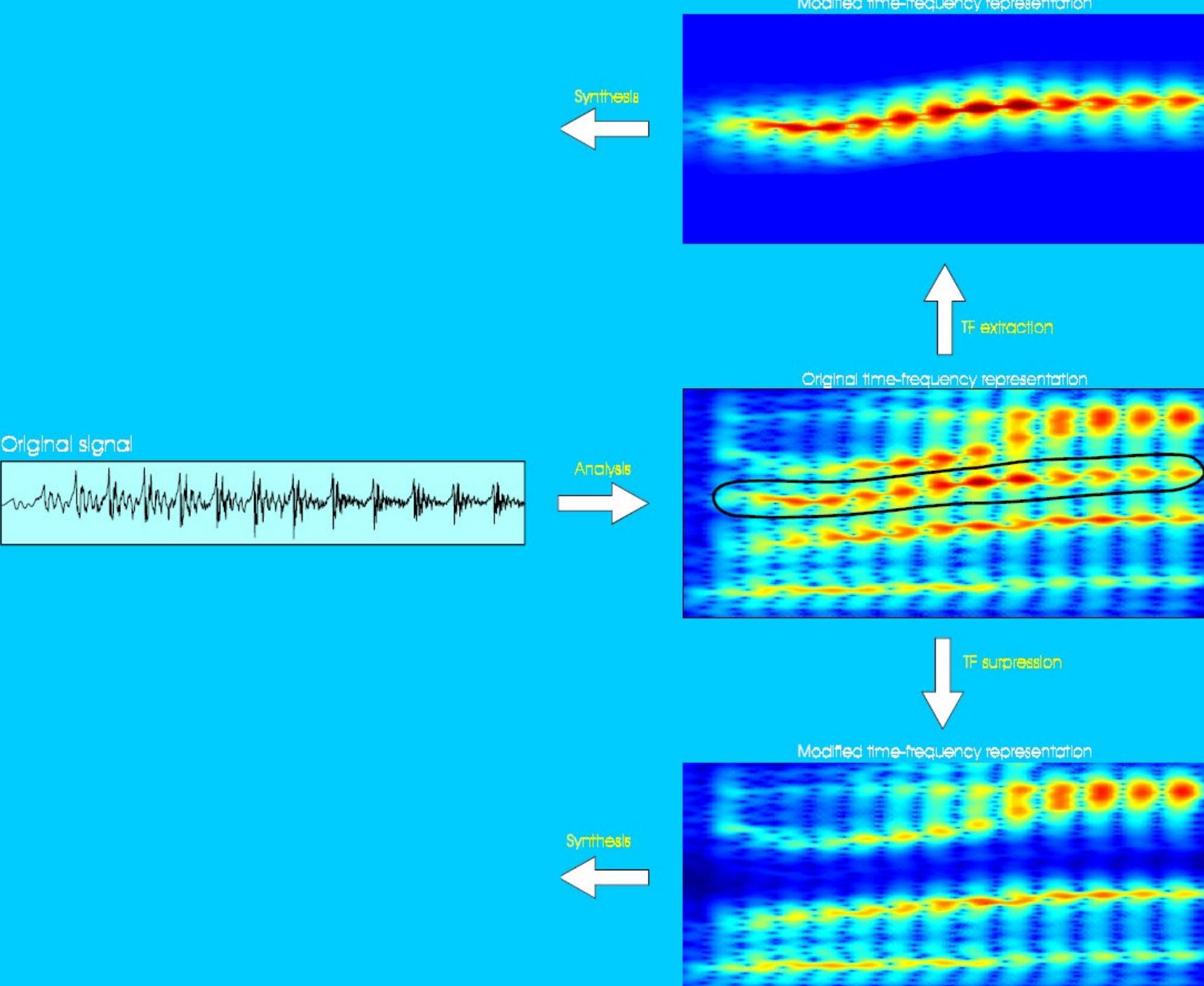
# STFT and ISTFT

- Note that the STFT is reversible by using the inverse STFT (ISTFT) if you keep the phase information i.e. you can reconstruct the original signal completely. Here 50% overlap is best.
- But if you have processed the information you may need to reconstruct/interpolate/extrapolate some phase information. For example: a phase vocoder.



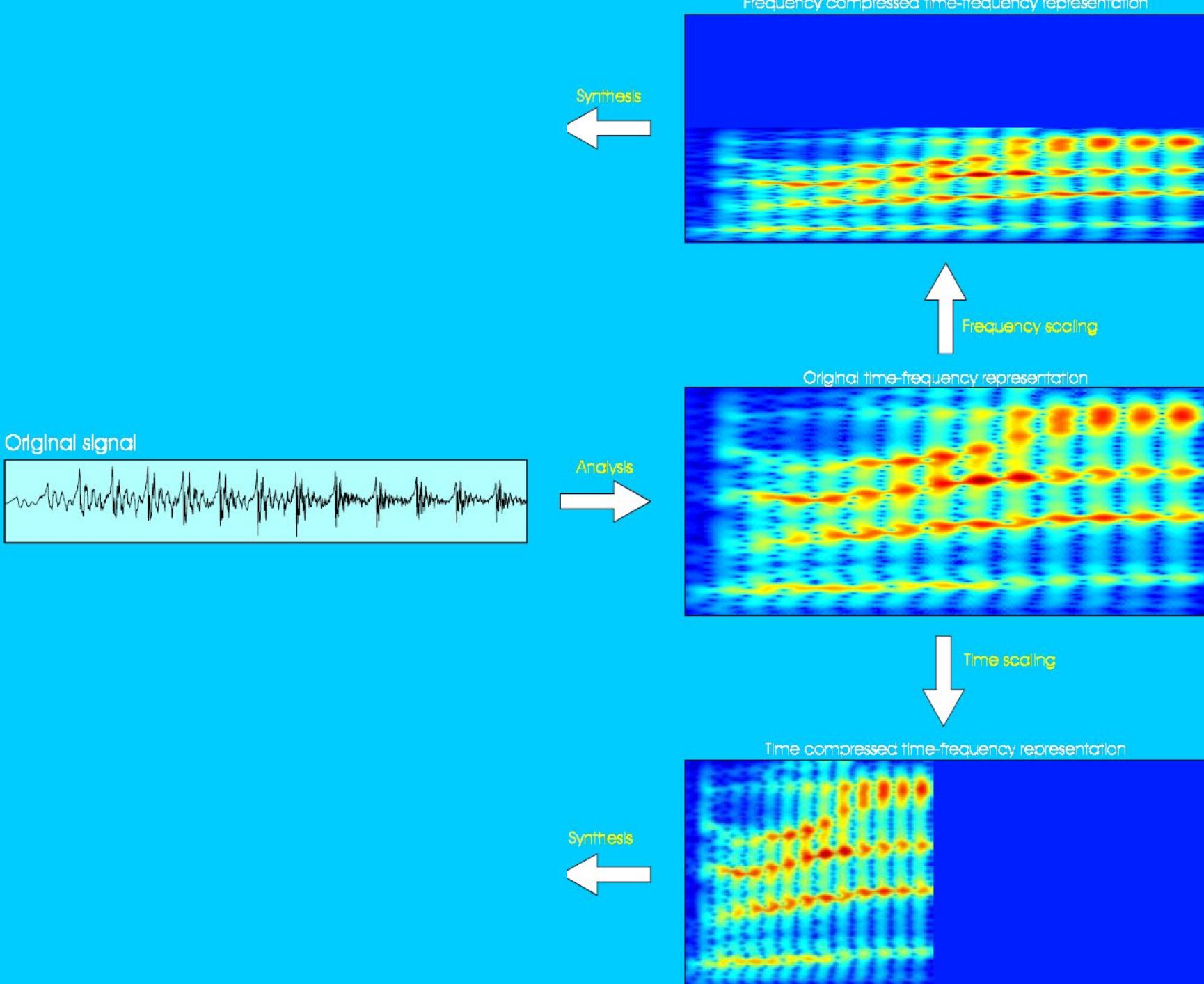
# Spectrogram Application example

- What happens here?



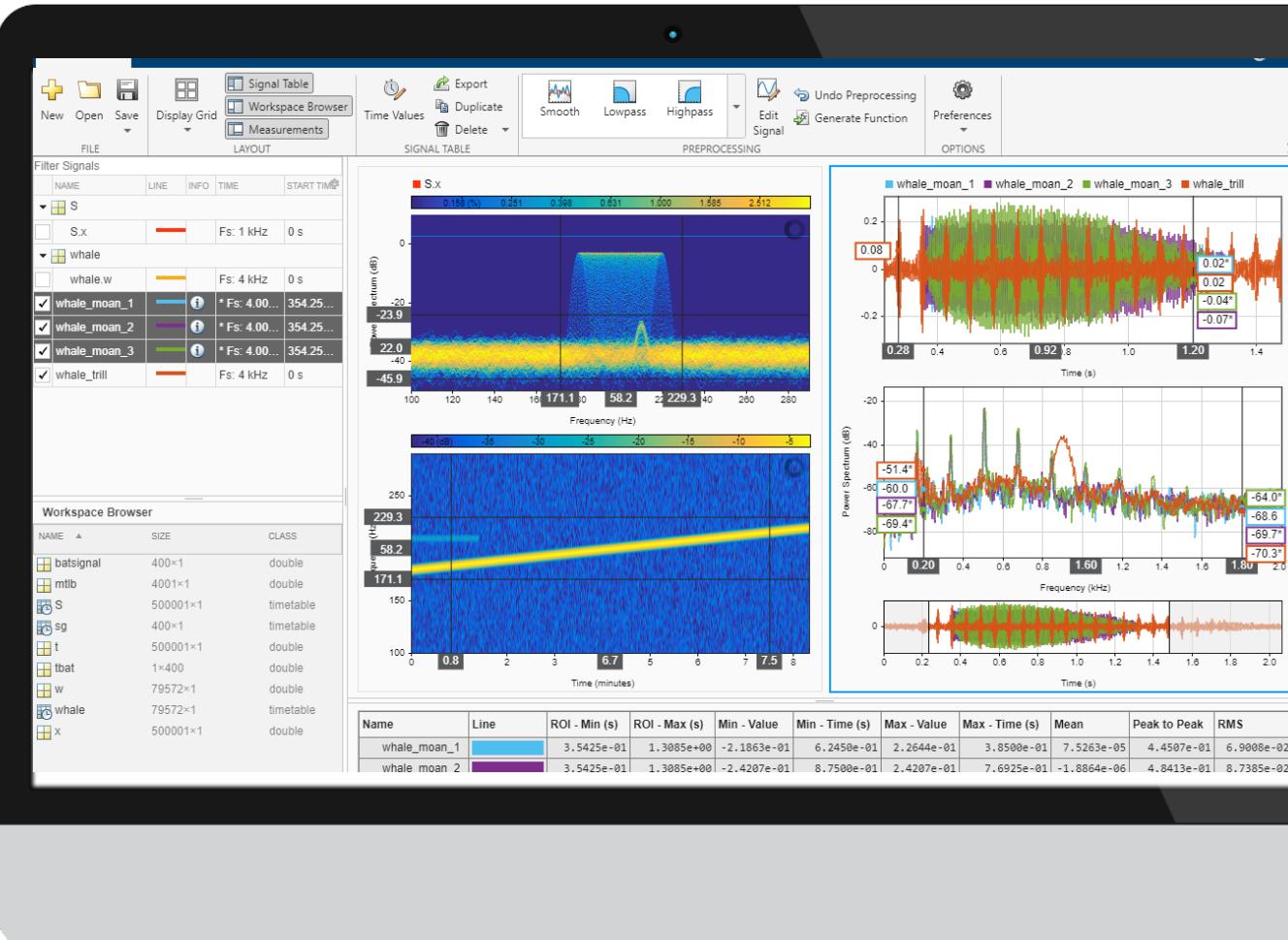
# Spectrogram Application example

- What happens here?



# Introducing the signalAnalyzer app in Matlab

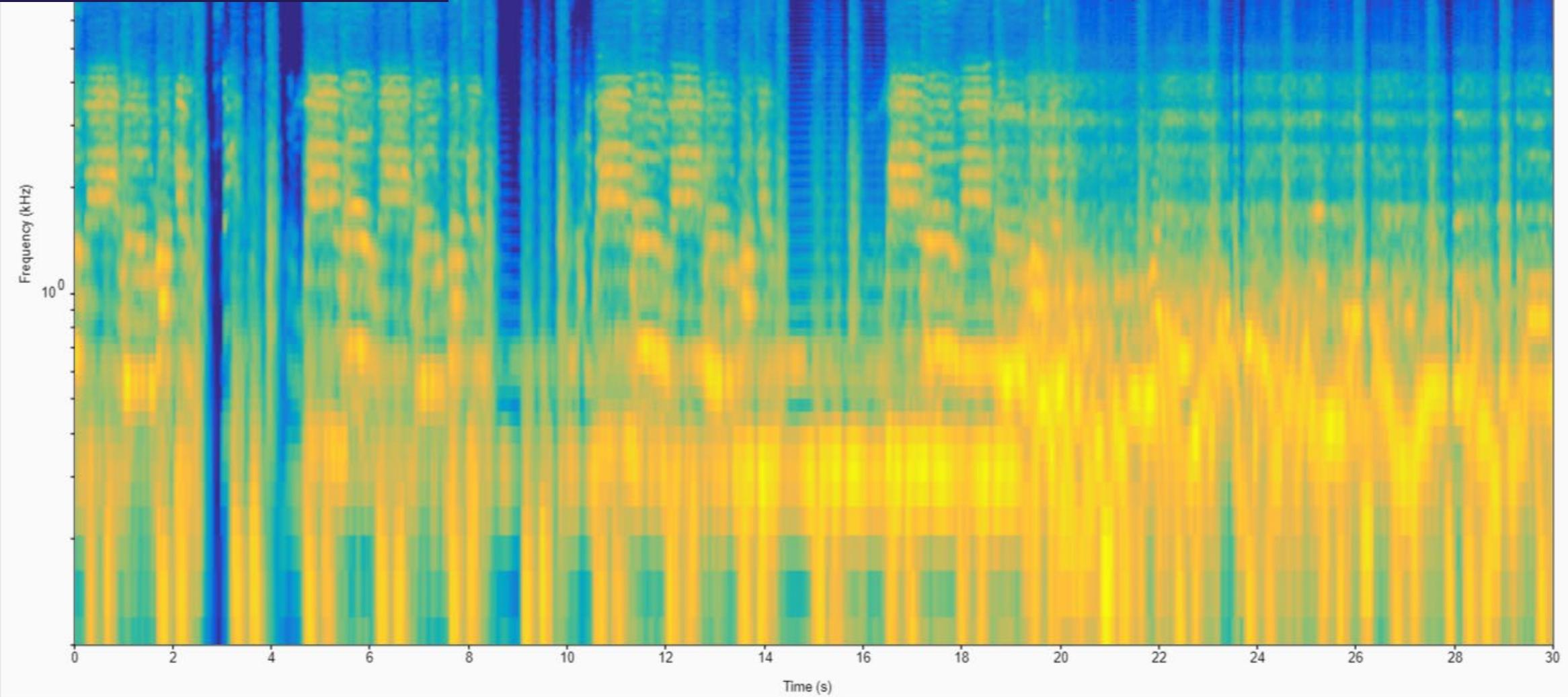
- ➊ Type: “signalAnalyzer” in Matlab in order to start the app.
- ➋ It is easy to import audio files directly from the work space in Matlab
- ➌ Filtering, and analysis of audio made easy 😊
- ➍ Notice that the spectrogram starts with automatic parameter choices – you will have to modify the parameters to make it really useful.  
E.g. set a lower limit to remove visual interference from unwanted noise in the signal.
- ➎ Also notice that the spectrogram features are rather limited – e.g. you cannot choose window type and it seems to ignore more extreme settings on e.g. window length.
- ➏ The Matlab function “spectrogram()” is much more flexible and can provide better resolution.



cutdata2

When does the guitar start?

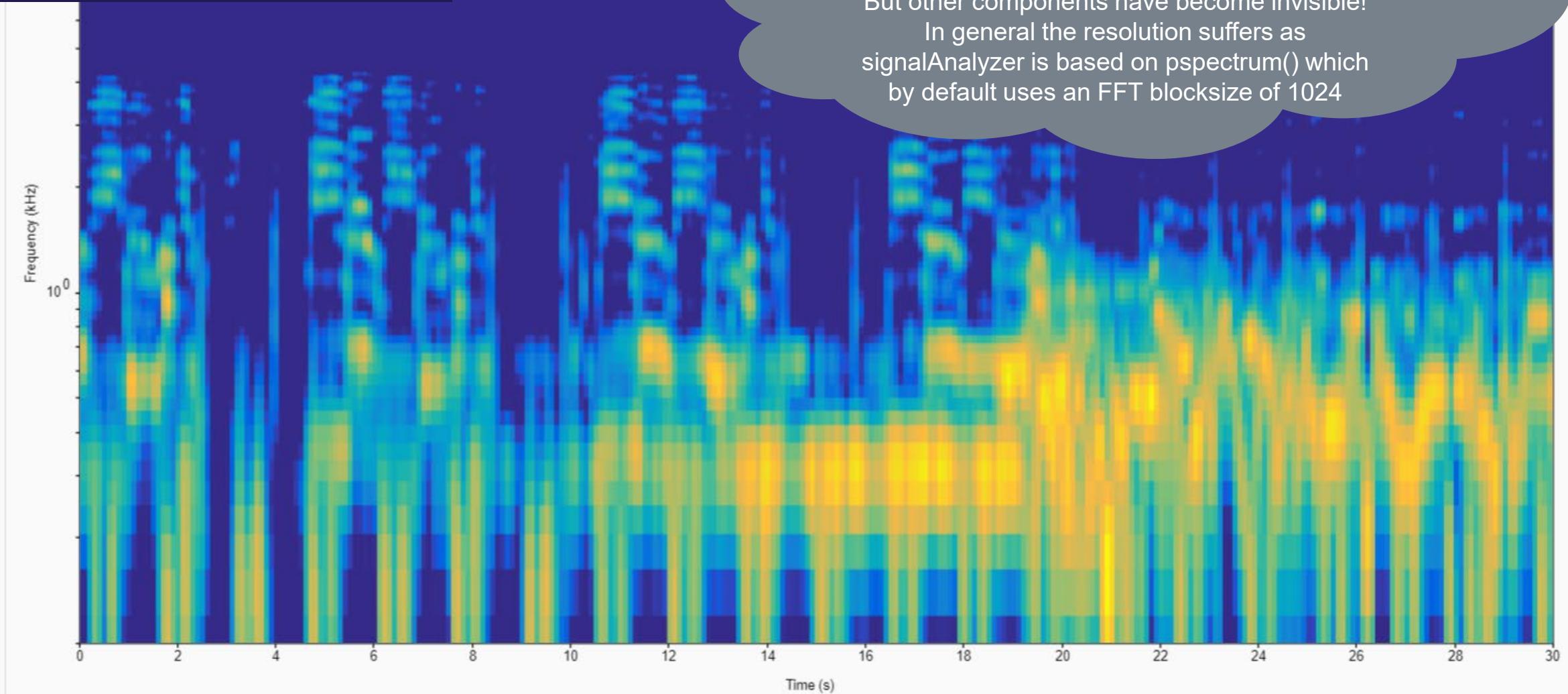
# Spectrogram Interpretation signalAnalyzer



cutdata2

When does the guitar start?

# Spectrogram Interpretation signalAnalyzer



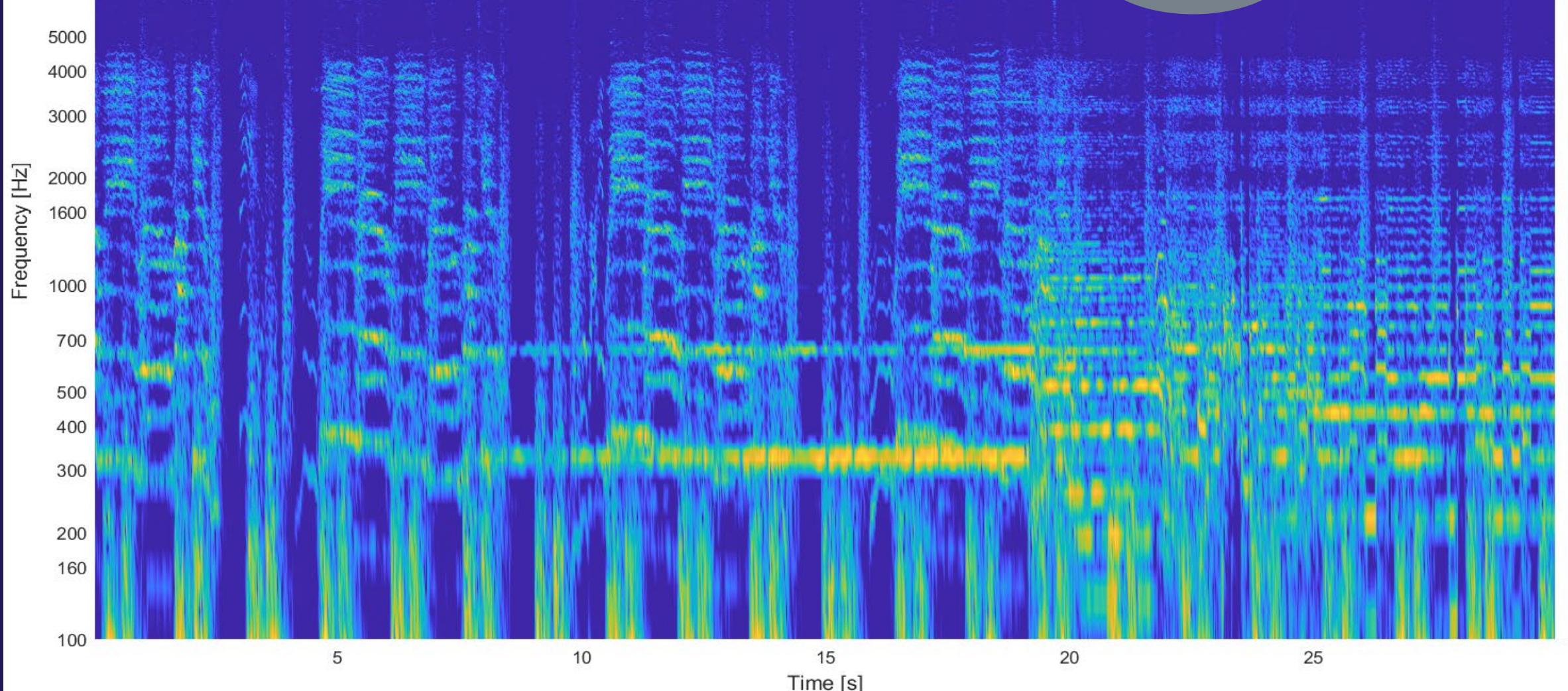
# Spectrogram Interpretation

## spectrogram()



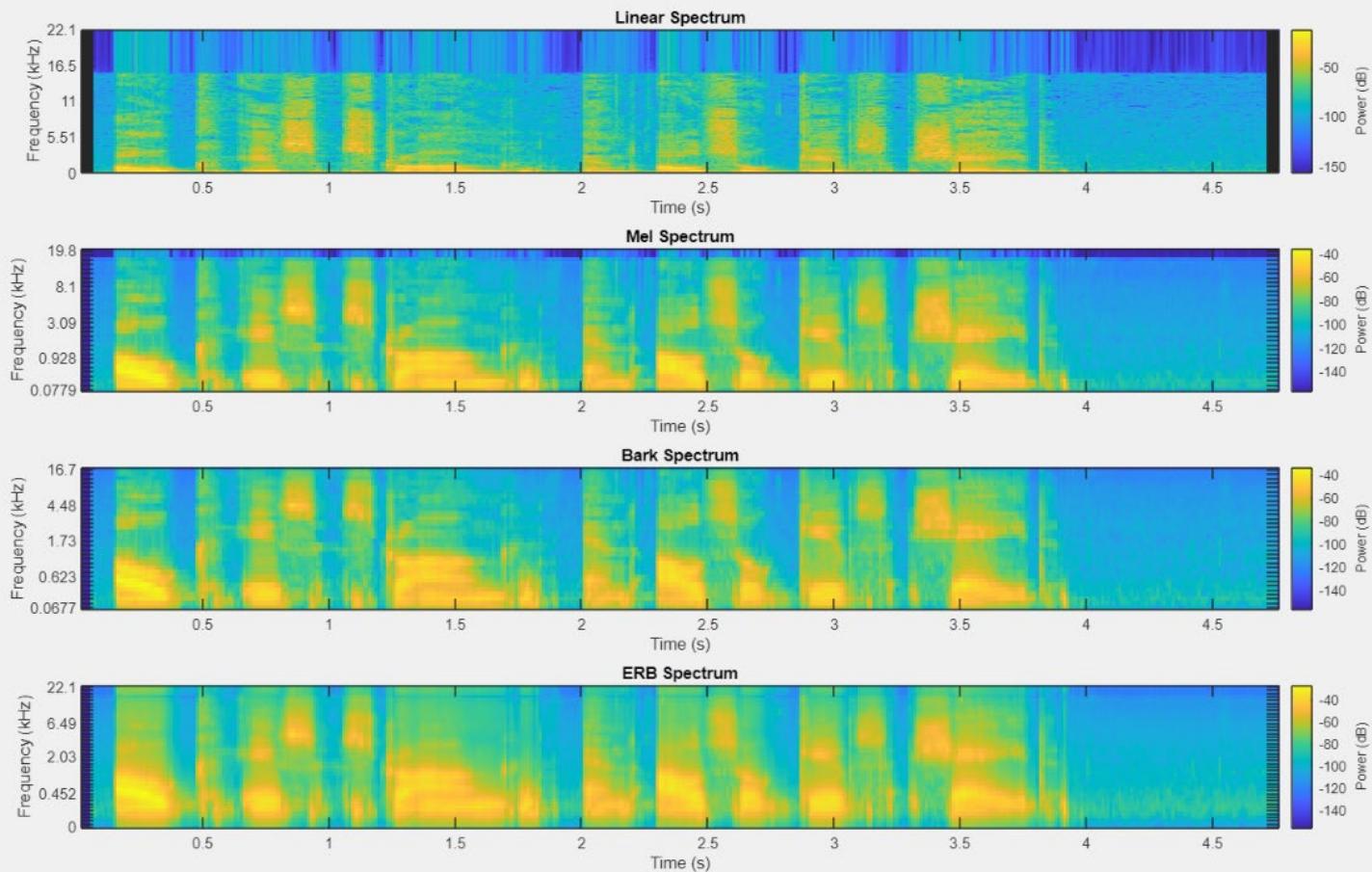
When does the guitar start?

This figure is based on the spectrogram function in Matlab  
Much more flexible but also more difficult to use – but notice the increased resolution in both time and frequency



# Take home message and next lecture

- ➊ The spectrogram is extremely useful for visualizing audio and understanding what happens over time.
- ➋ The frequency scale is linear while our perception is logarithmic – this means that modifications can be made to better reflect our perception – and possibly we can throw away some data making it even more useful for AI and large data sets.
- ➌ This will be one of the topics next time, where we will explore audio features commonly used for AI.
- ➍ We will try to get a hands on feeling for what each feature is useful for and how to extract them from the audio signal



# Exercises

- Start by using the Matlab app signalAnalyzer to analyze the provided signal – “Exercise1.wav”. First try the spectrum, then the spectrogram,
- For more detailed time-frequency analysis make a Matlab script that uses the spectrogram function in Matlab. Here you must define your own window – e.g. hann window, overlap size etc.  
Try to 3D rotate the plot to get a feel for the 3D information stored in the spectrogram related to the color information stored in the 2D image depending on your color range.
- For Problem 3 - Try to estimate the fundamental frequencies – these are related to our perceived pitch and therefore the melody you hear.
- I will provide some Matlab code solutions tomorrow morning.

Problem 1: What frequencies does Exercise1.wav contain

Problem 2: How does the frequencies change over time? (Analyze Exercise1.wav with the spectrogram function). Note the timing and frequency content.

Problem 3: Analyze Exercise3.wav with the spectrogram function. Can you find the same frequencies as in Exercise1.wav?

Problem 4: Find a speech signal, a music signal and an environmental sound signal. Analyze with the spectrogram function – try to link what you hear with what you see.

Examine the influence of changing the window size and explain it. First with window size 256 then windows size 1024 and finally 4096





# Group rooms (for those not on CE8-AVS??)

#?



AALBORG UNIVERSITY  
DENMARK

**THE END**