6.300 Signal Processing

Week 9, Lecture A:
Short Time Fourier Transform

- Spectrograms
- Overlap-add (streaming applications)

Quiz 2: Thursday November 7, 2-4pm 50-340
- Closed book except for two pages of notes (8.5'' x 11'' both sides)
- No electronic devices (No headphones, cell phones, calculators, ...)
- Coverage up to Week #8 (DFT)
- practice quiz as a study aid, no HW # 9

Time-varying Signals

Real-world signals (i.e., speech, music, . . .) often have frequency content that varies with time.

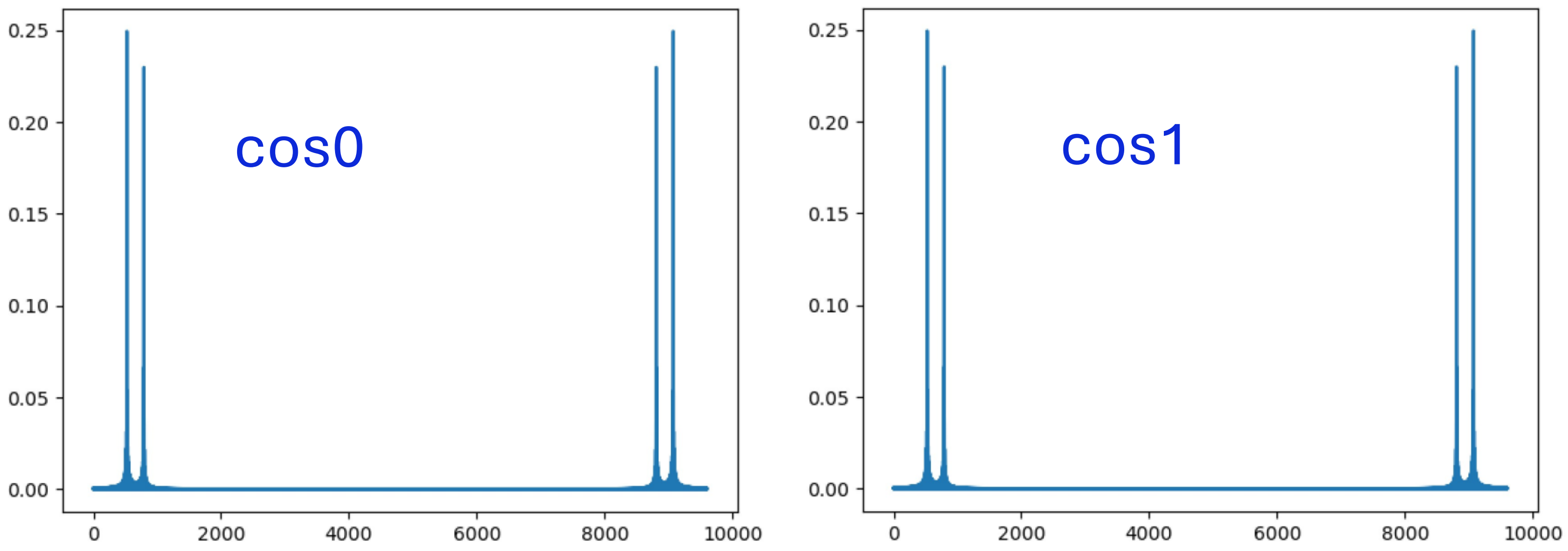
Fourier Transform: events that are local in time are global in frequency (and vice versa). Sudden changes and local variations can be difficult to detect.

Example: 2 tunes

- cos0.wav
- cos1.wav

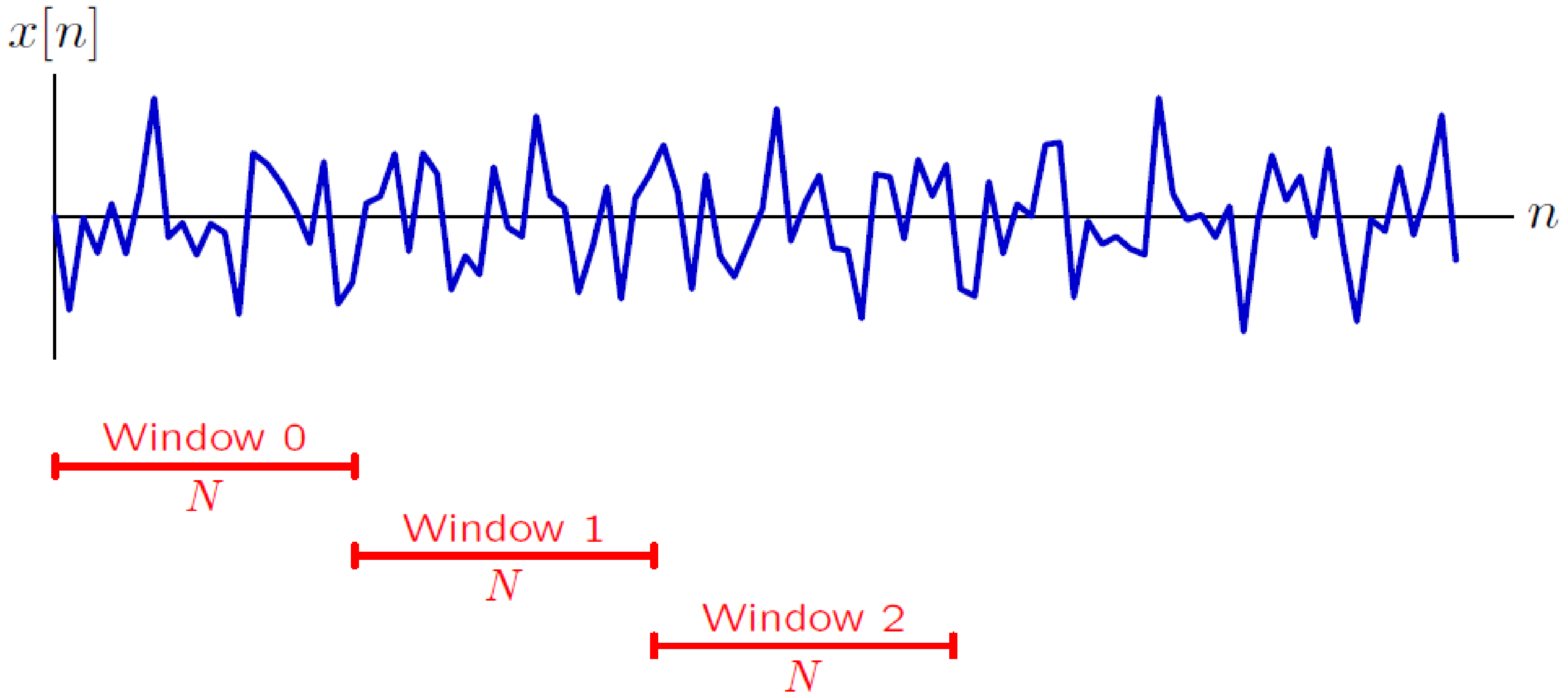
Can we tell them apart?

FFT of the two signals:



Short-Time Fourier Transform (STFT)

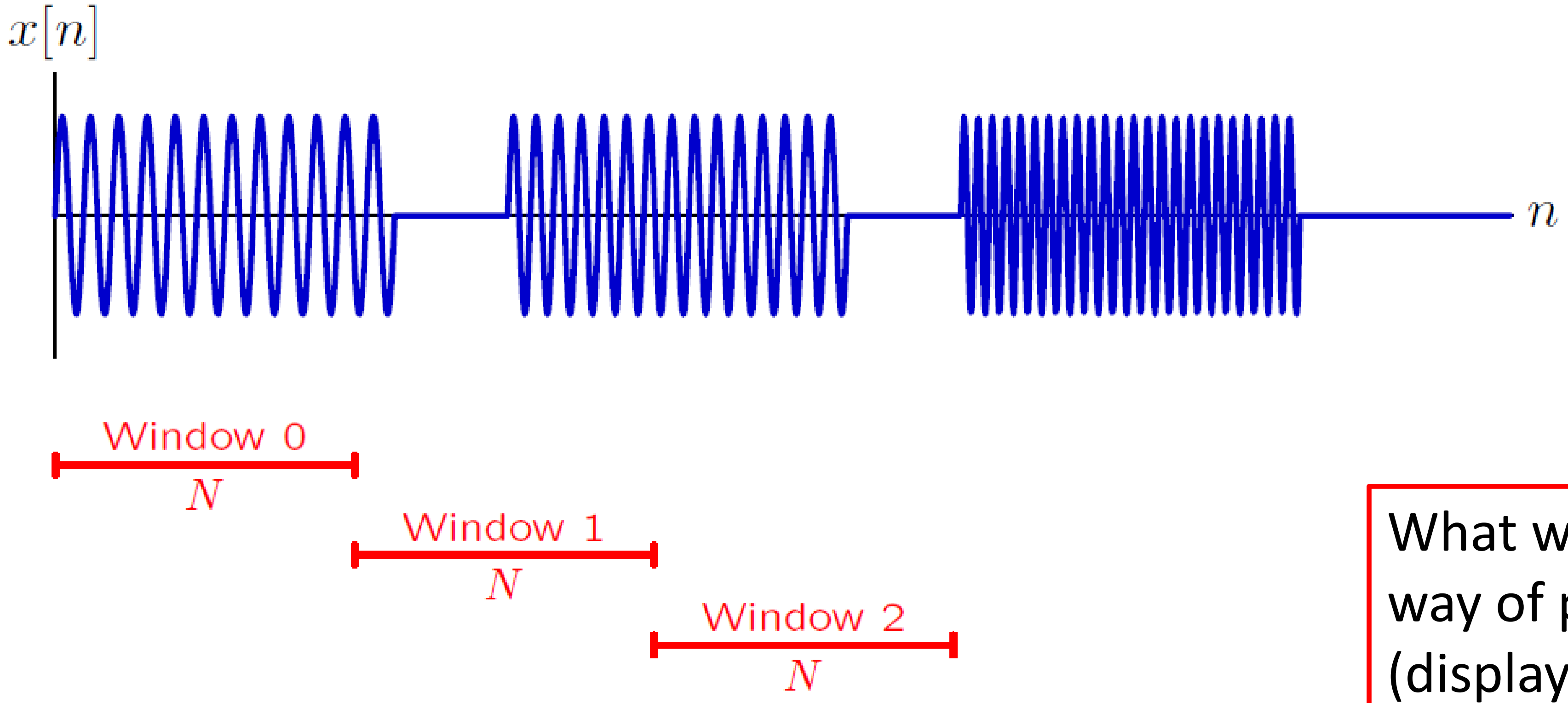
Short-time Fourier transforms (STFTs) represent the frequency content of a long signal by that of a sequence of shorter DFTs.



- Each DFT is computed for a time interval of length N.
- Successive time intervals begin at increasingly later times.

Short-Time Fourier Transform (STFT)

Each window highlights frequencies from a different part of time.



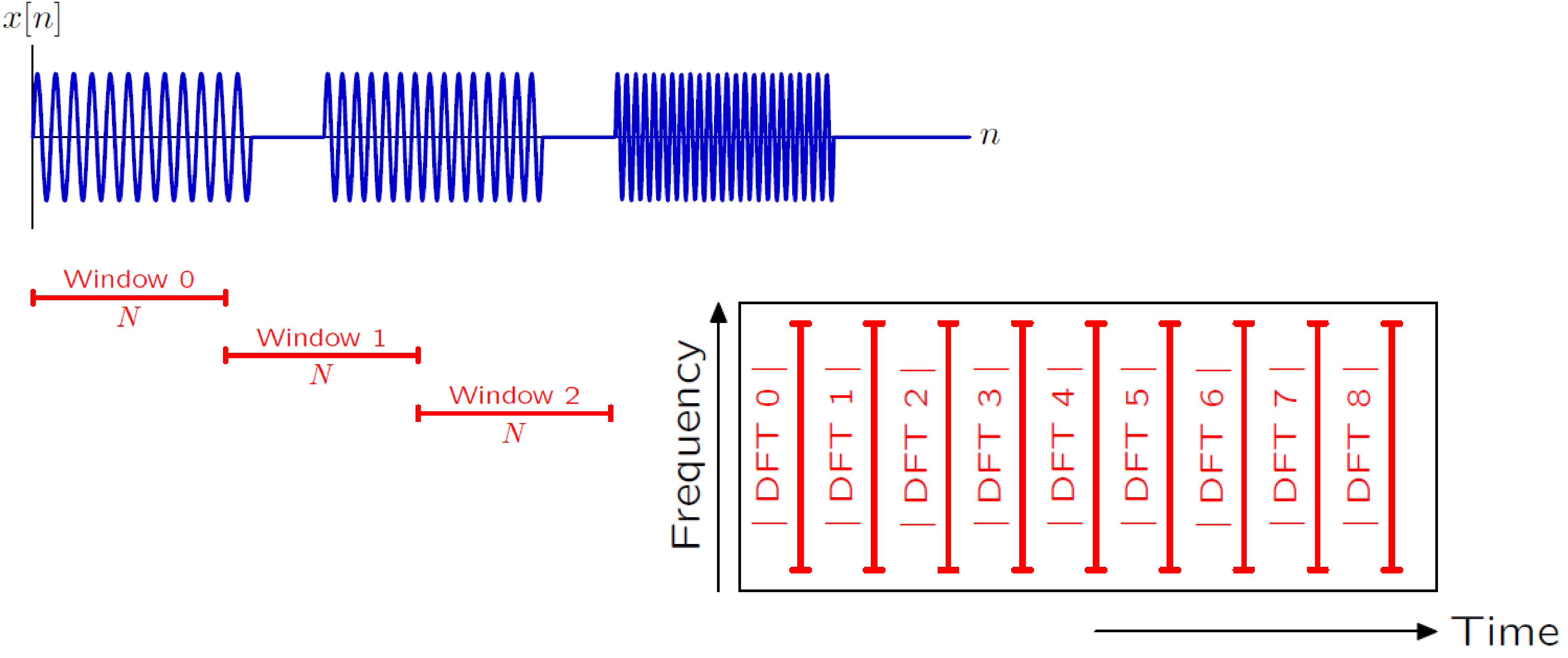
What will be a good way of presenting (displaying) these DFTs?

Window 0 highlights the frequency of the first tone.
Window 1 contains contributions from the first two tones.

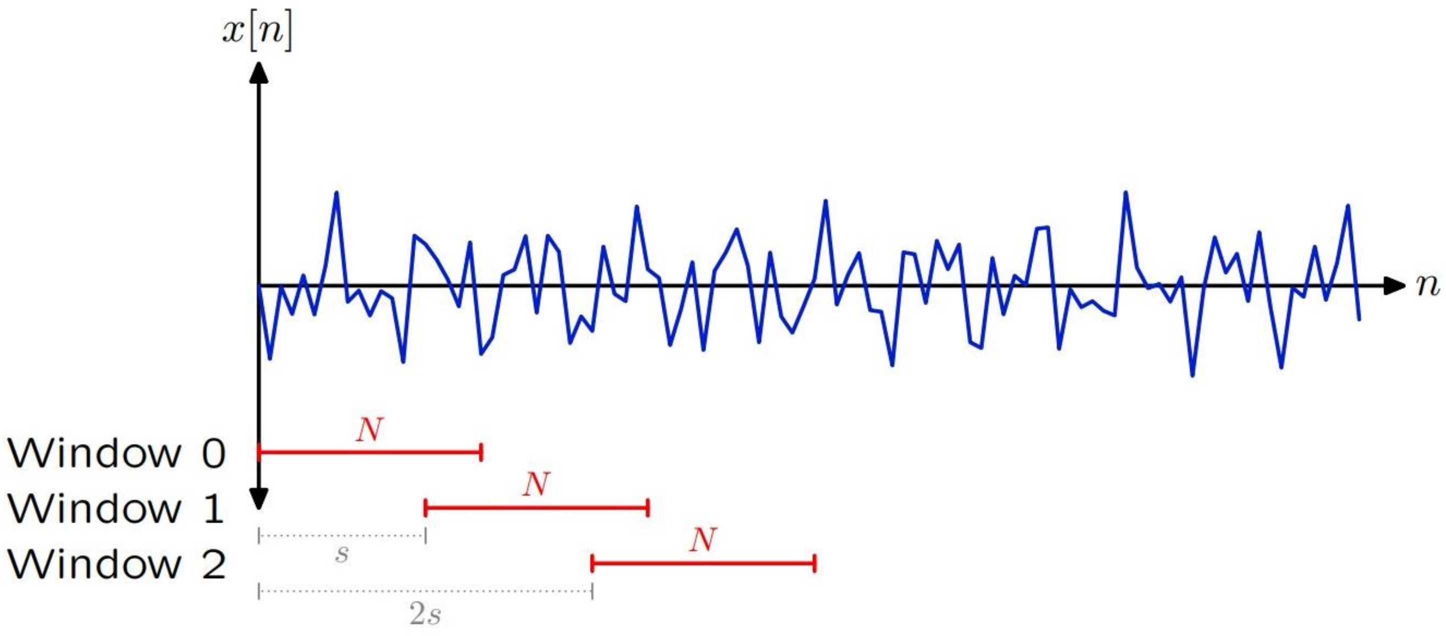
Window 2 ....

Spectrogram

A spectrogram displays the successive DFT magnitudes as columns in a two-dimensional representation.



STFT and Spectrograms



Where:

- m is a time index (window nymber, k is a frequency index
- N is the length of a window
- s is “step size”
- $w[n]$ is window function

Formally, we define the STFT of a signal x as:

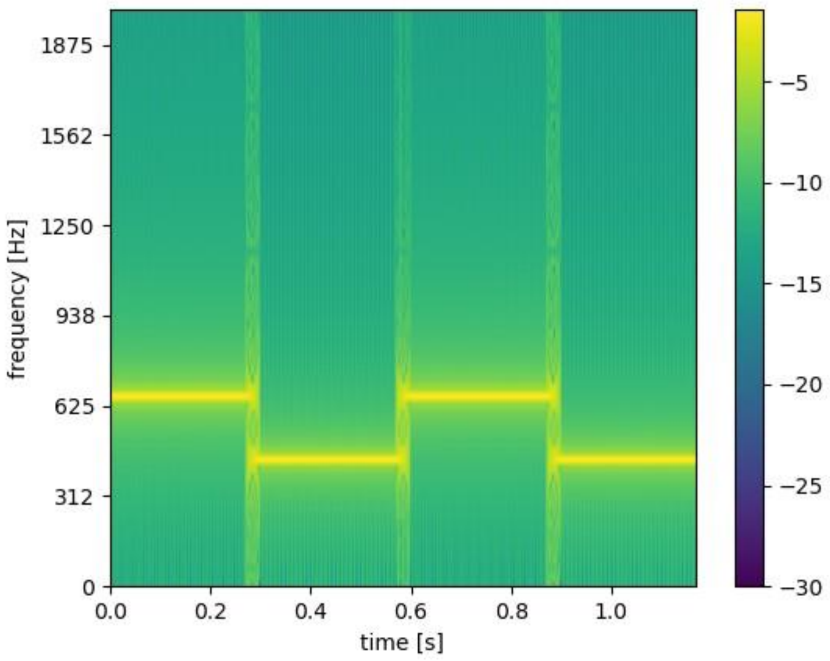
$$STFT\{x\}[m,k] = \sum_{n=0}^{N-1} x[n + m \times s] \cdot w[n] e^{-j\frac{2\pi k}{N}n}$$

STFT and Spectrograms

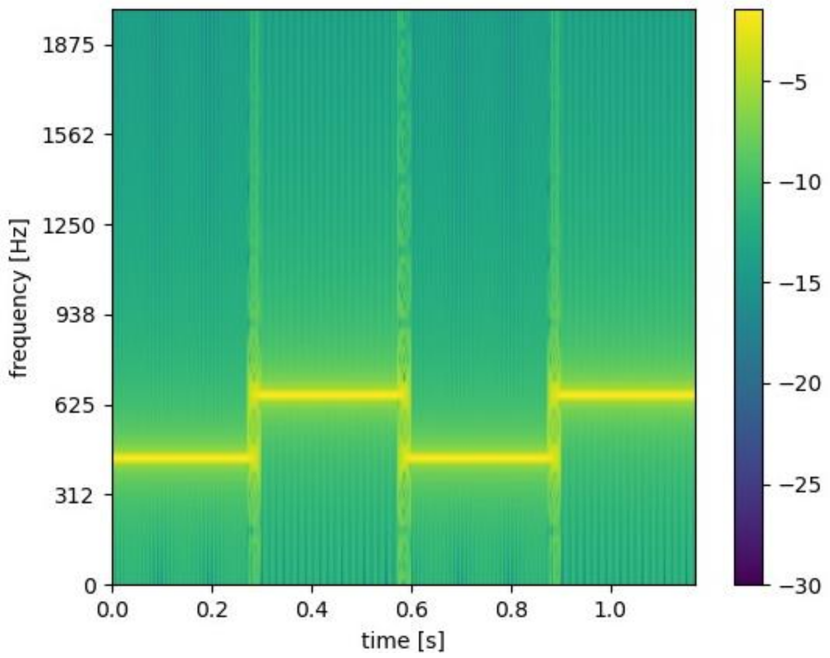
$$STFT\{x\}[m,k] = \sum_{n=0}^{N-1} x[n + m \times s] \cdot w[n] e^{-j\frac{2\pi k}{N}n}$$

The STFT is often visualized using a spectrogram, which is defined to be the magnitude squared of the STFT.

The following images show spectrograms for the previous tone sequences.



Window_size=256, step_size=16, cos0.wav

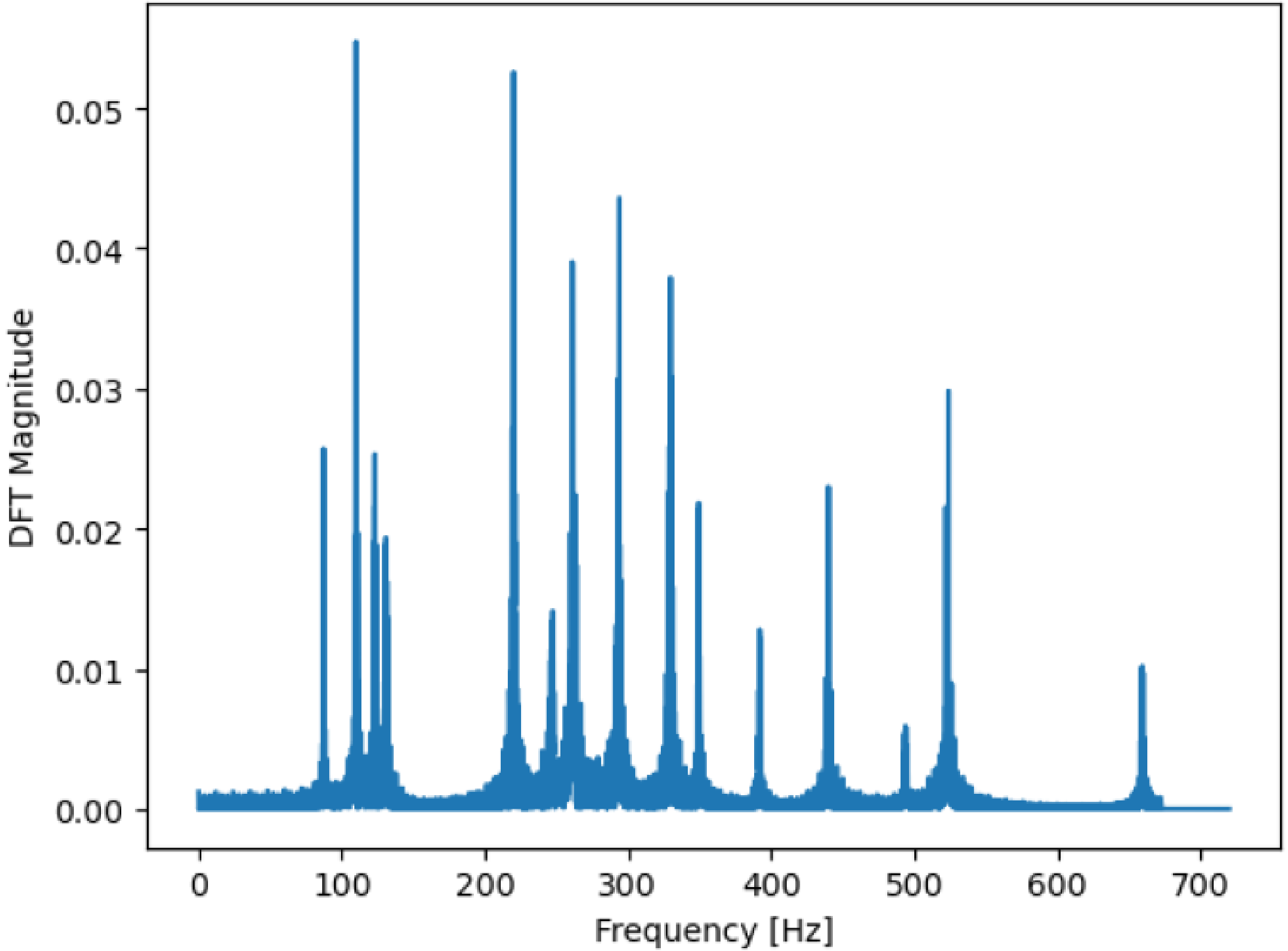


Window_size=256, step_size=32, cos1.wav

We can now clearly tell the difference between the two signals.

Example 2: Music Clip

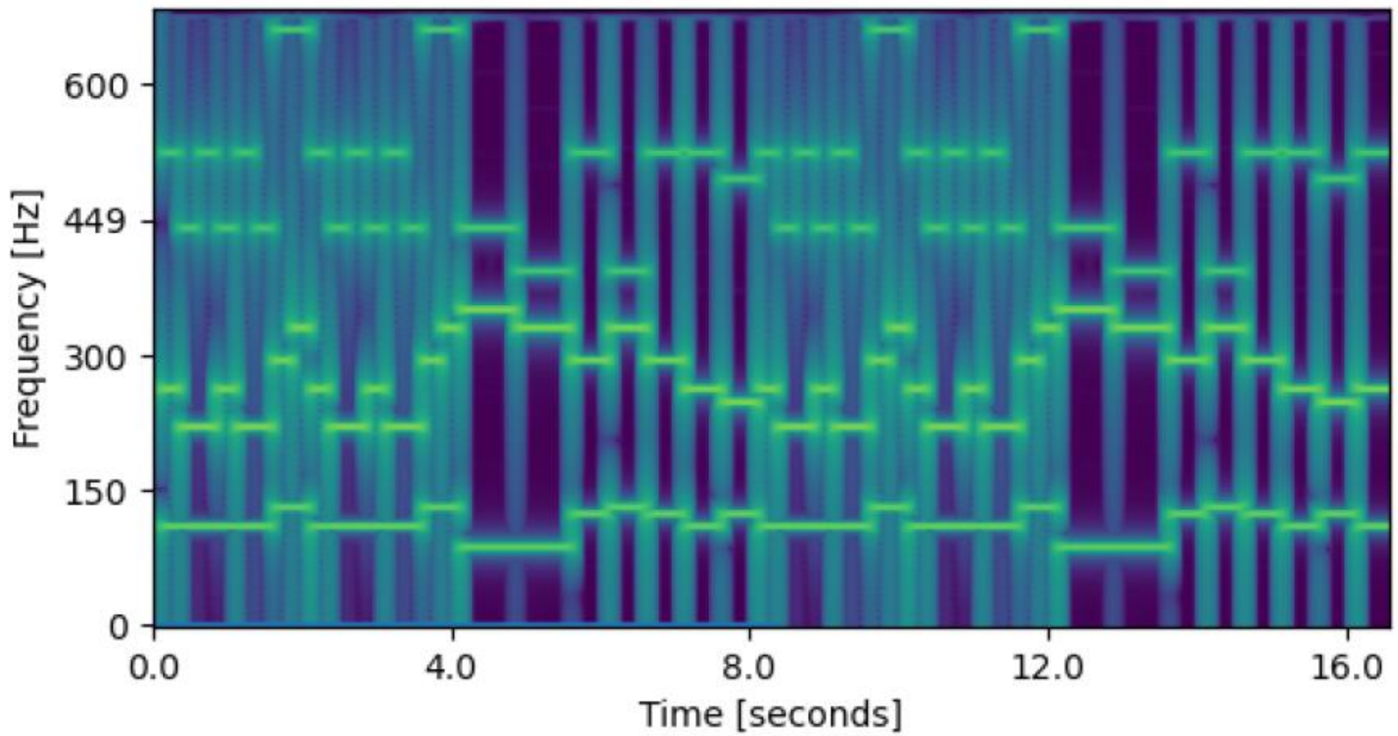
This plot shows the magnitudes of DFT coefficients for a clip of music.



Can you identify the piece?

Spectrogram

The spectrogram shows three parts: bass, melody, and harmony.



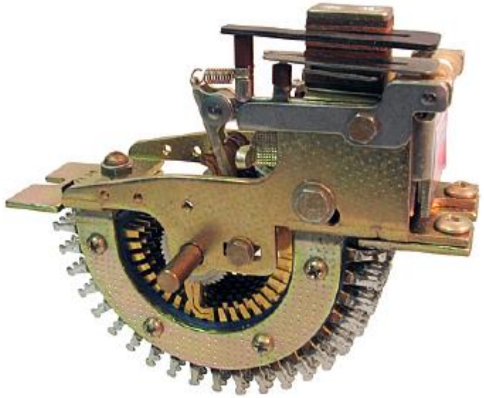
This score was created from the spectrogram using Lilypond (GNU project).

Structure is similar to that of musical score.

Example 3: Spectrograms in Telephony

In early telephone systems (circa 1880) clients were connected manually through a switchboard.

The first automated systems (circa 1900) used pulses from a rotary dial to route calls via complicated systems of relays.



Dual Tone Multi Frequency Signaling

In the 1950s, Bell Labs pioneered a system to route phone calls using tones.

		High-Group Frequencies			
		1209Hz	1336Hz	1477Hz	1633Hz
Low-Group Frequencies	697Hz	1	ABC 2	DEF 3	A
	770Hz	GHI 4	JKL 5	MNO 6	B
	852Hz	PRS 7	TUV 8	WXY 9	C
	941Hz	*	OPER 0	#	D

This method is still in use today, especially as a way to collecting data:
“Please enter your 16-digit account number followed by the # key."

Dual Tone Multi Frequency Signaling

In the 1950s, Bell Labs pioneered a system to route phone calls using tones.

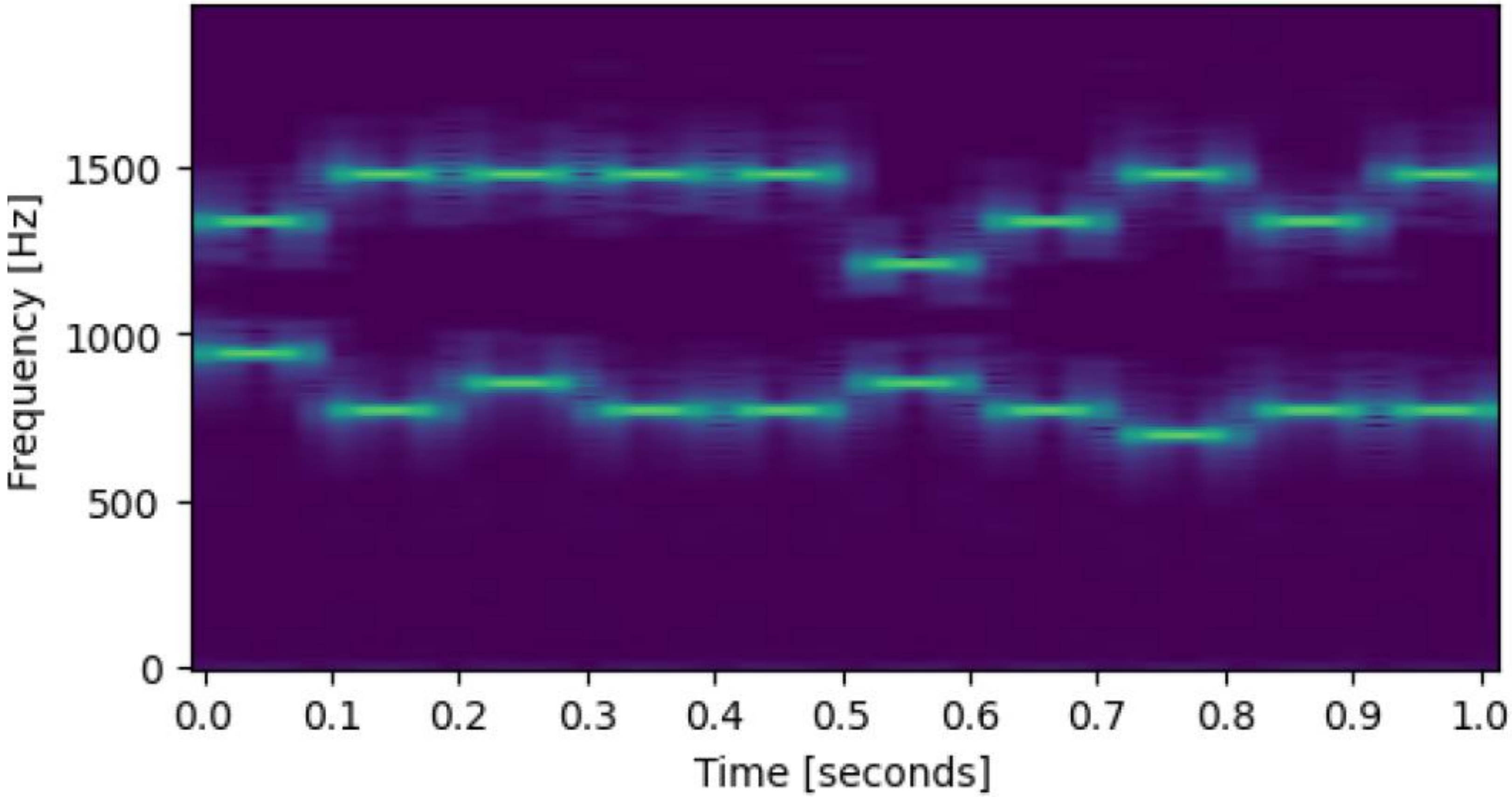
		High-Group Frequencies			
		1209Hz	1336Hz	1477Hz	1633Hz
Low-Group Frequencies	697Hz	1	ABC 2	DEF 3	A
	770Hz	GHI 4	JKL 5	MNO 6	B
	852Hz	PRS 7	TUV 8	WXY 9	C
	941Hz	*	OPER 0	#	D

Pressing a button transmits two frequencies: one representing the row number and one representing the column number (the last column is rarely used today).

Decoding the sequence of tones requires recognizing those two frequencies.

Spectrogram

Here is a spectrogram for a DTMF signal.

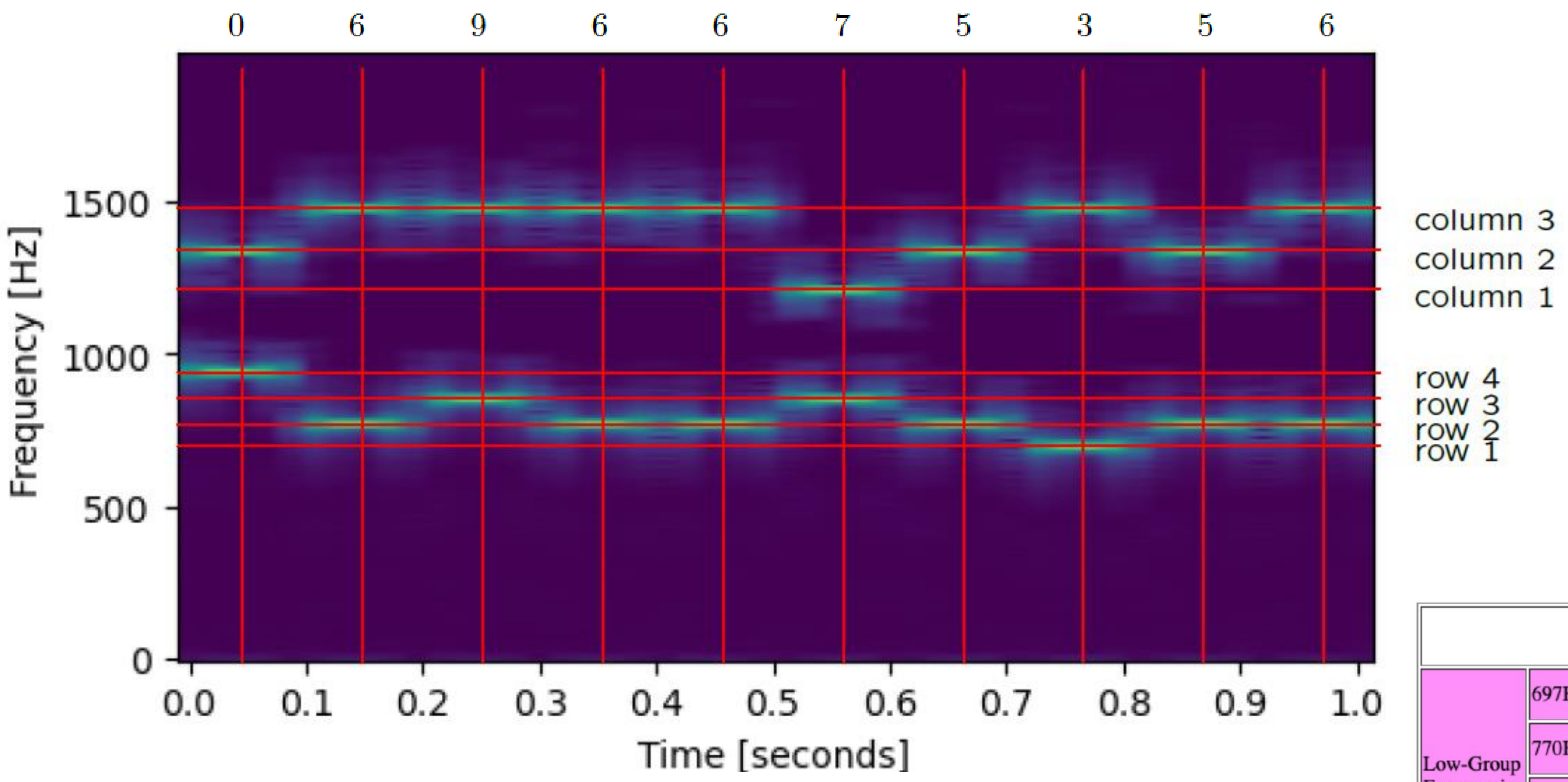


Can we find out what is this code?

It clearly shows a sequence of frequency pairs.

Spectrogram

Here is a spectrogram for a DTMF signal.



The pair uniquely identifies the pushbutton number.

		High-Group Frequencies			
		1209Hz	1336Hz	1477Hz	1633Hz
Low-Group Frequencies	697Hz	1	ABC 2	DEF 3	A
	770Hz	GHI 4	JKL 5	MNO 6	B
	852Hz	PRS 7	TUV 8	WXY 9	C
	941Hz	*	OPER 0	#	D

Check yourself

Choosing parameters for a spectrogram.

Determine values of N that can be used to construct a spectrogram to separate musical notes based on DFT analysis of a recording at 44,100 samples/sec.

Errors in frequency should be less than or equal to 5 Hz, so that we can resolve middle C (261.63 Hz) from C# (277.18 Hz).

Frequencies should be computed separately for 1/8 second windows, so that we can tell if two notes are sounding together or separately.

What DFT analysis length N will work best?

1. $N > 4410$
2. $N < 4410$
3. $N < 5512$
4. $4410 < N < 5512$
5. none of the above

Consider here there is no overlap between the windows

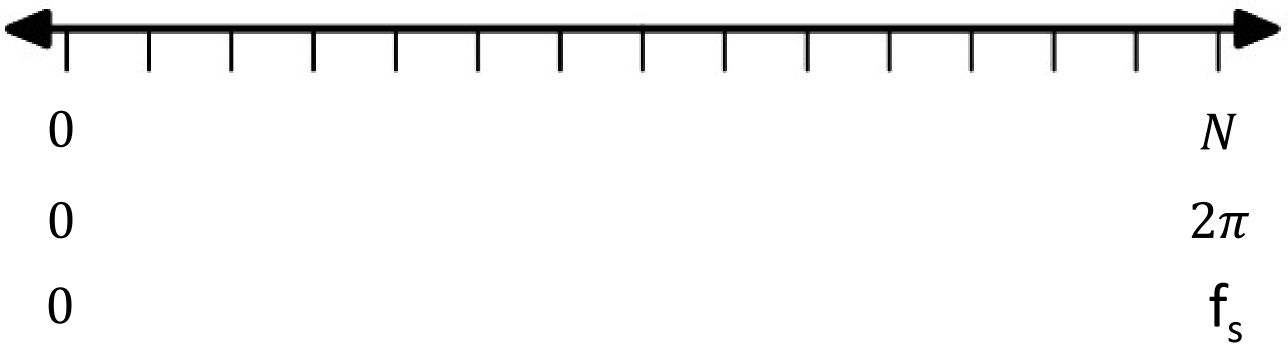
Check yourself

Choosing parameters for a spectrogram.

To make frequency errors less than 5 Hz, we need to analyze frequencies into bins of ** width, or smaller. ⬅ 10 Hz

A DFT breaks the full range of frequencies (44,100 Hz) into N bins, so the bin width is **, and this bin width should be less than 10 Hz.
So N $\geq$ **. ⬅ 44100/N, 4410

A DFT breaks time into chunks of length **. To keep the chunks smaller than 1/8 second, **< 1/8 so N should be less than **. ⬅ N/f_s, N/f_s, 5512



k: integer (frequency)
Ω: rad/sample
f: cycles/second (Hz)

Check yourself

Choosing parameters for a spectrogram.

Determine values of N that can be used to construct a spectrogram to separate musical notes based on DFT analysis of a recording at 44,100 samples/sec.
Errors in frequency should be less than or equal to 5 Hz, so that we can resolve middle C (261.63 Hz) from C# (277.18 Hz).
Frequencies should be computed separately for 1/8 second windows, so that we can tell if two notes are sounding together or separately.

What DFT analysis length N will work best? 4

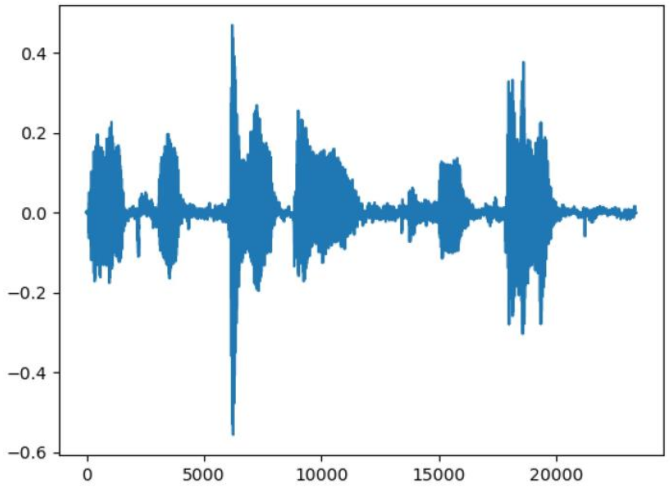
1. $N > 4410$
2. $N < 4410$
3. $N < 5512$
4. $4410 < N < 5512$
5. none of the above

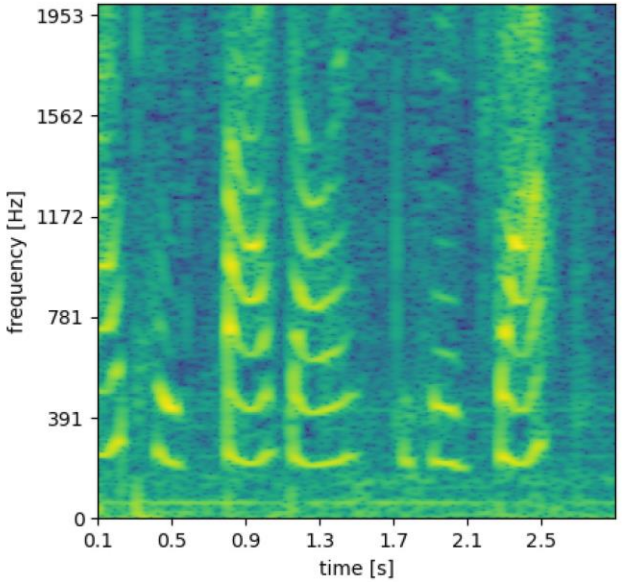
Short-Time Fourier Transform

Short-Time Fourier transforms are useful for constructing spectrograms, to visualize the frequency content of a signal as a function of time.

- convert important information into a single picture, process as image.

- next week, we will look at how spectrograms are used in analyzing speech.



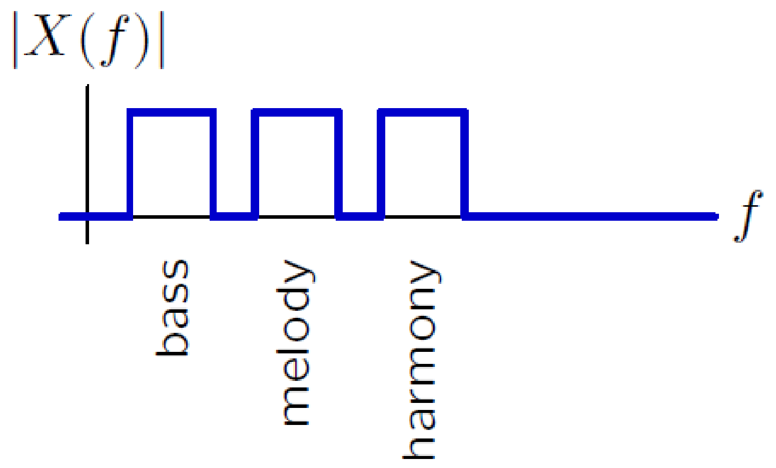


Short-Time Fourier transforms are also useful for processing long signals, such as those that are common in streaming applications.

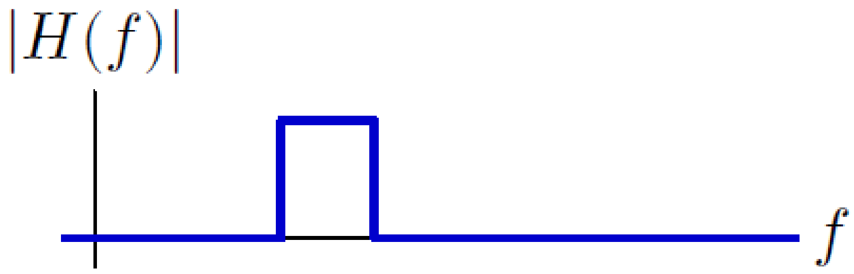
Example

Consider a musical piece that contains three simultaneous “voices," each playing a single sinusoidal tone.

Voice 1 (bass):	40-170 Hz
Voice 2 (melody):	170-340 Hz
Voice 3 (harmony):	340-750 Hz



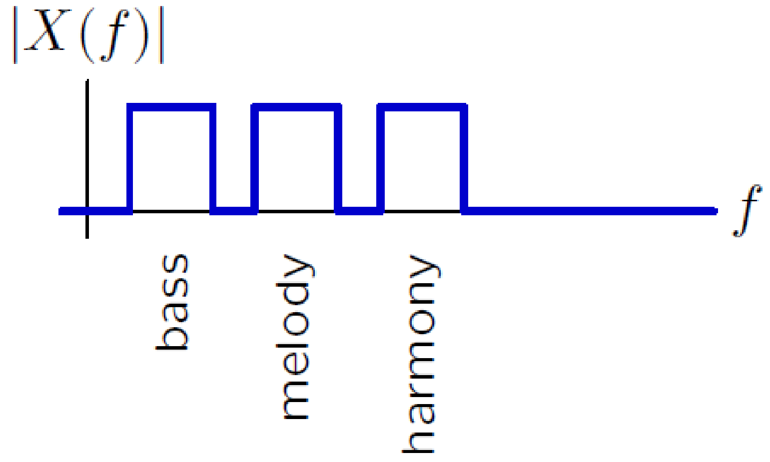
We would like to remove the bass and harmony voices, leaving just melody.



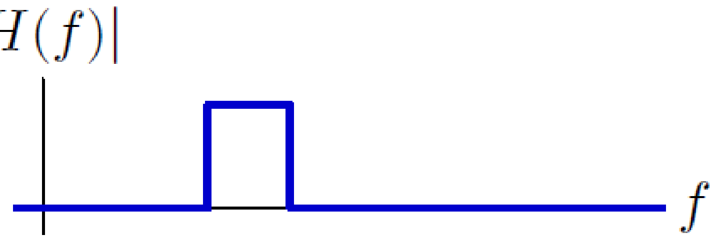
Example

Consider a musical piece that contains three simultaneous “voices," each playing a single sinusoidal tone.

Voice 1 (bass): 40-170 Hz
Voice 2 (melody): 170-340 Hz
Voice 3 (harmony): 340-750 Hz



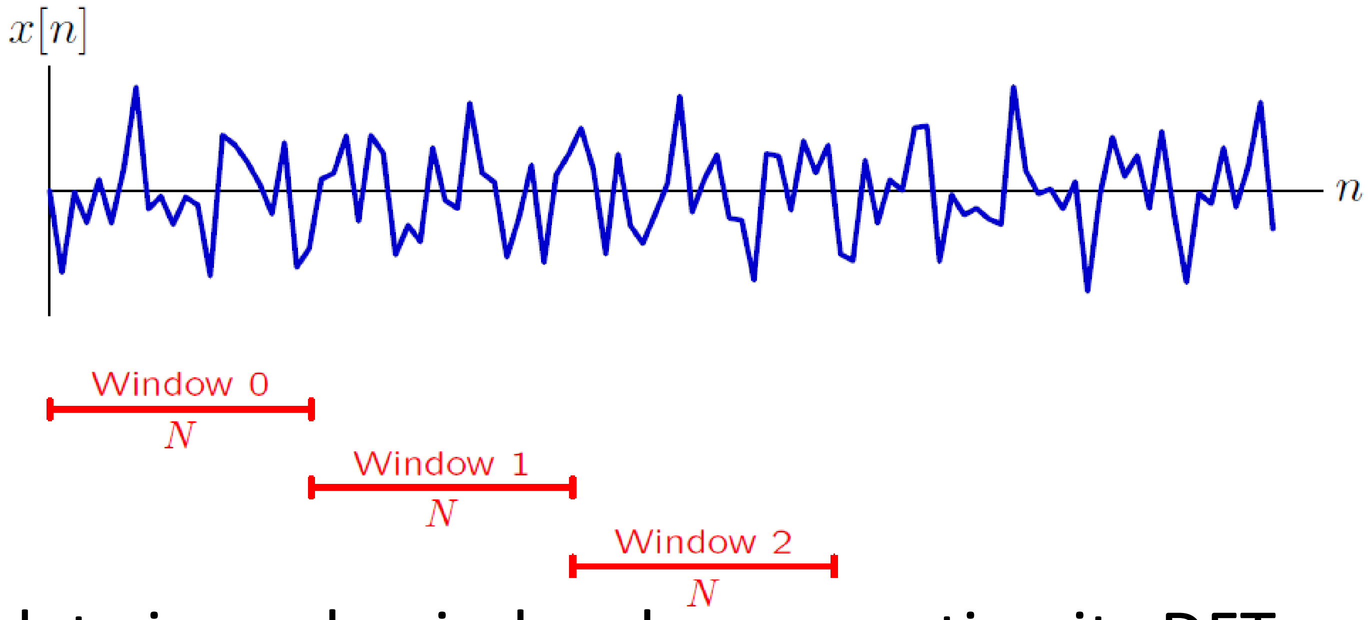
We would like to remove the bass and harmony voices, leaving just melody.



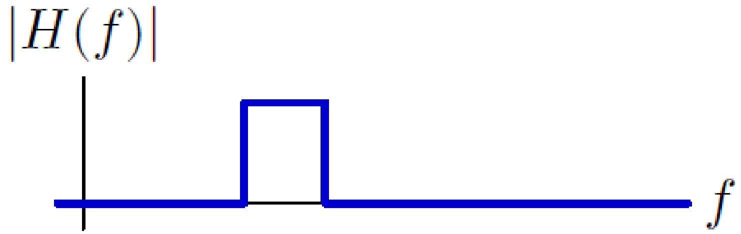
The straightforward approach is to filter the DFT of the piece, passing only the frequencies of interest. This straightforward approach requires accessing the entire piece before any part is ready to play. This approach is problematic for streaming applications.

Streaming Algorithm 1

Divide a signal *x*[*n*] into a sequence of shorter signals of length *N*.



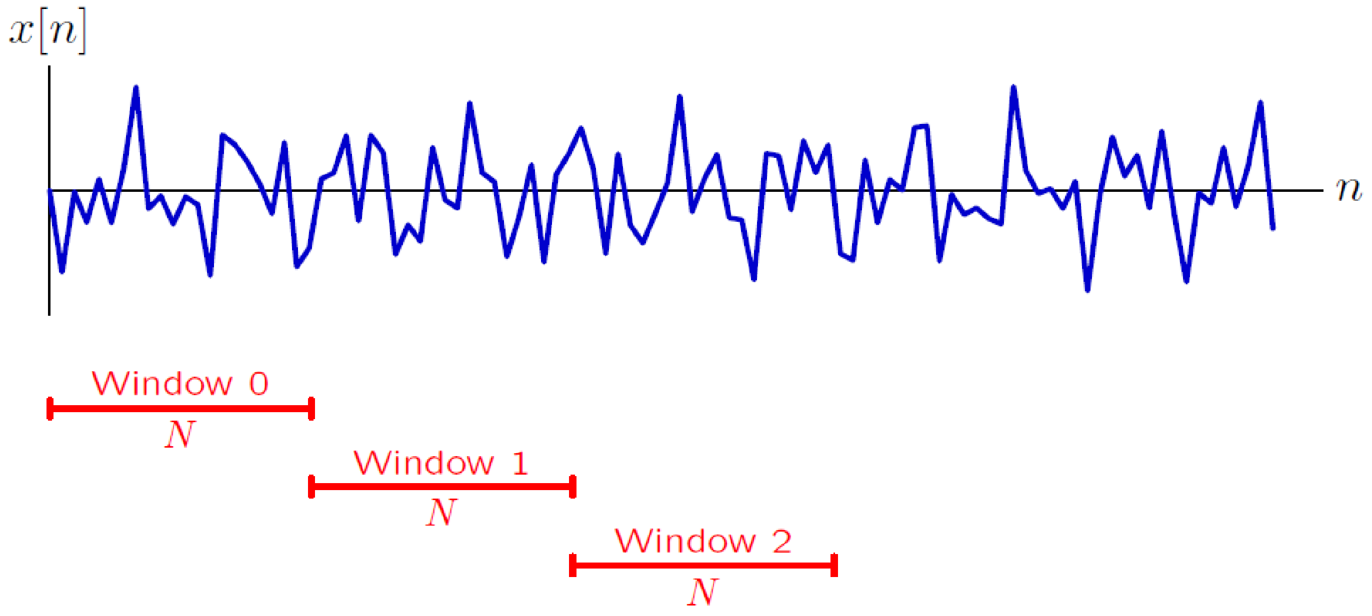
Filter data in each window by computing its DFT and zeroing components outside passband.



Assemble results for each window to form the output signal.

How Effective is Algorithm 1?

Divide a signal *x*[*n*] into a sequence of shorter signals of length *N.*



Compare the **original:**

- am_resynth.wav

Where does the clicks come from? What is the problem for this method?

participation question

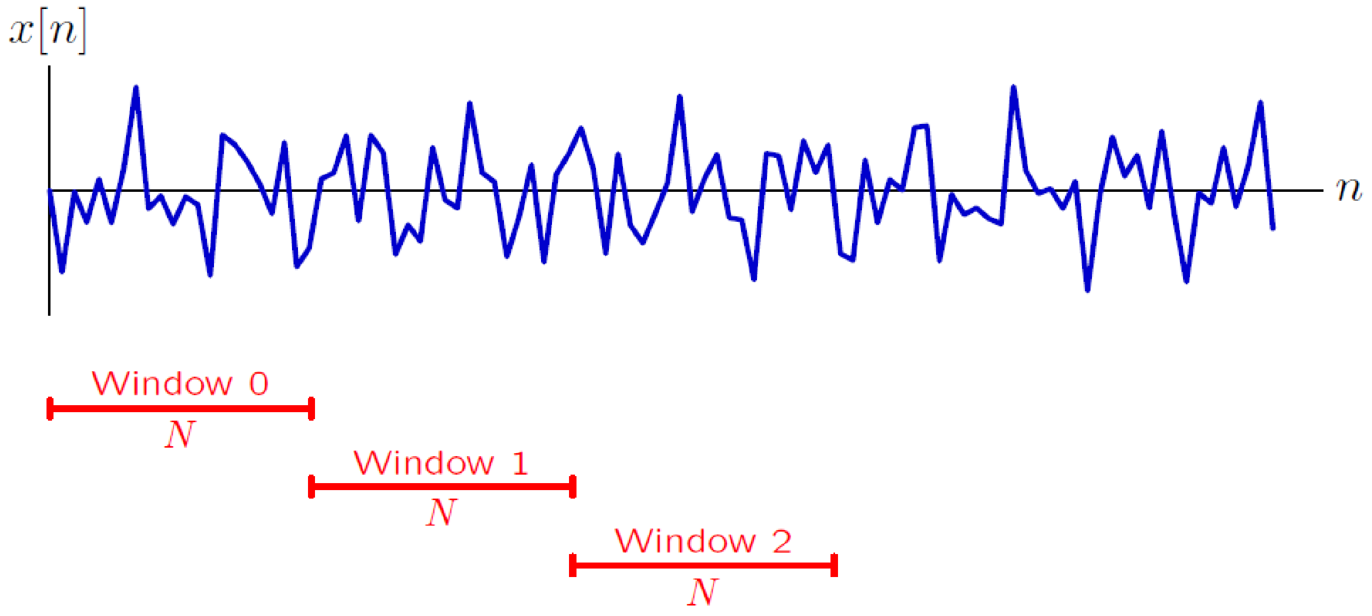
with a version that is processed **one window at a time:**

- am_algorithm1.wav

It isolated the melody, but also added clicks!

How Effective is Algorithm 1?

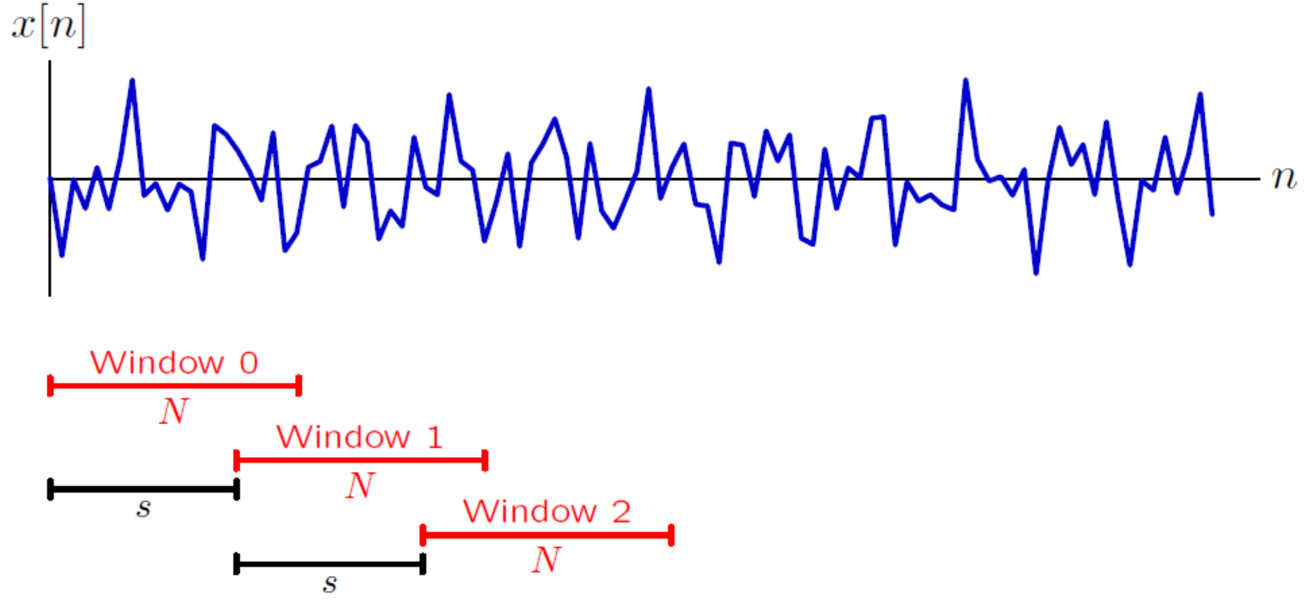
There are at least two major problems with this approach.



- The length of (x * h)[n] is generally > length of x[n] or h[n]. Part of result from each window should fall into an adjacent window(s).
- Even worse, the convolution will be circular if implemented with a DFT. Results from window 1 that should fall into window 2 will alias back to the beginning of window 1!

Overlap-Add Method

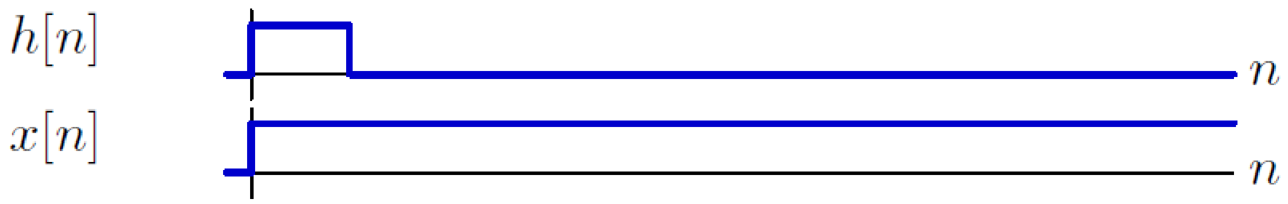
Avoid circular convolution artifacts and spill over problems by filling each window with just s < N input samples and then zero-padding.



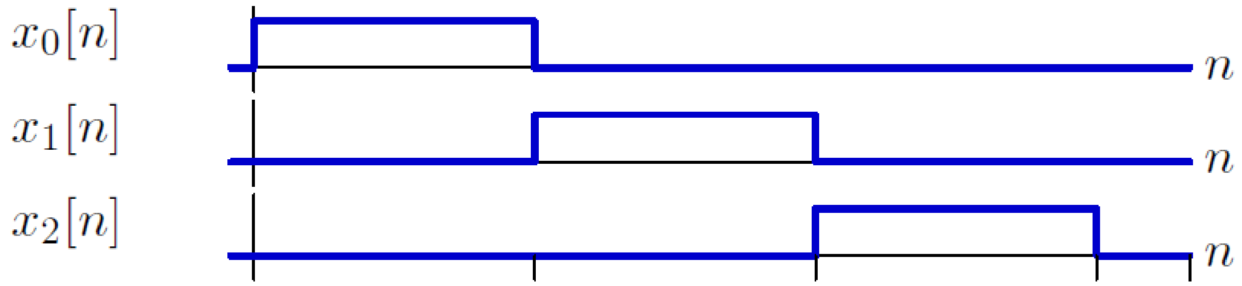
- If the length of h[n] ≤ N−s + 1, then the length of h[n] convolved with s samples of the input will be less than N => no circular convolution artifact.
- If the length of h[n] ≤ N−s + 1, then the overlapping portions of adjacent windows will accommodate spill over between windows.

Overlap-Add: Graphical Depiction

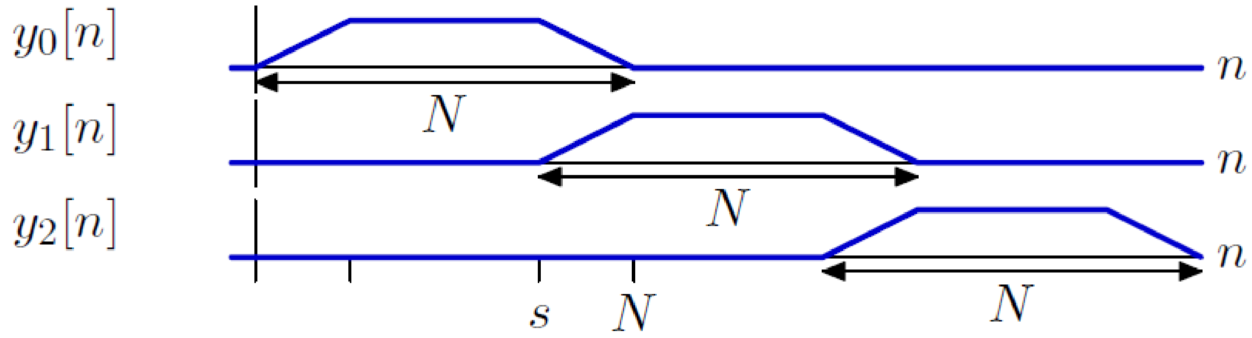
Convolve a square pulse with a signal that is 1 for all n.



Divide the input $x[n]$ into pieces that are each of length s.



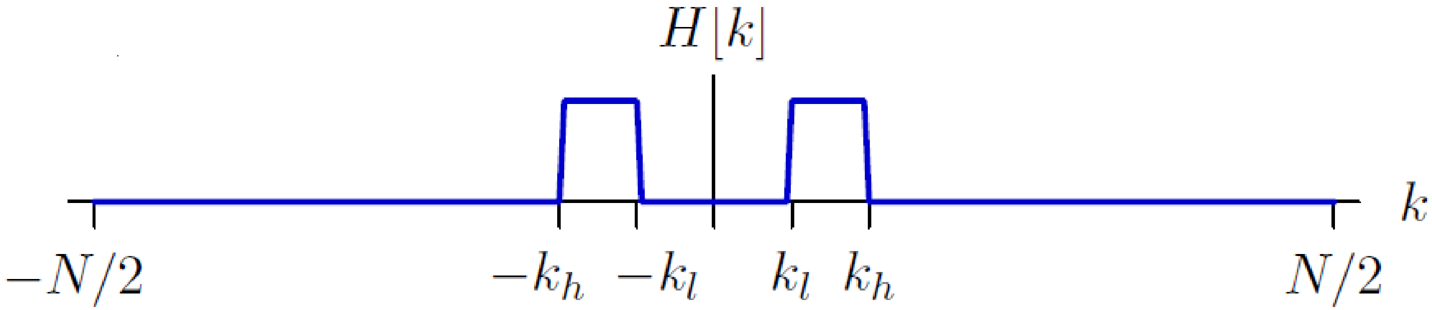
Convolve each piece of $x[n]$ with $h[n]$.



Then the output $y[n] = y_0[n] + y_1[n] + y_2[n] + \cdots$ Hence overlap-add.

Filter Design

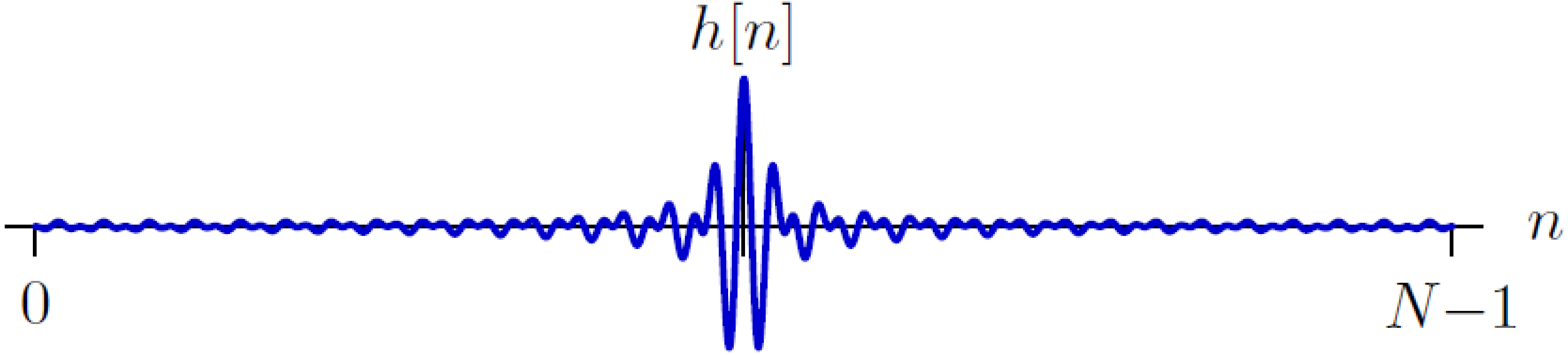
Design a filter to isolate the melody using the overlap-add method.



The filter should pass frequencies in the range $f_l \leq f \leq f_h$.

$$k_l = \frac{f_l}{f_s} N \qquad k_h = \frac{f_h}{f_s} N$$

If we take the window length N = 8192, then h[n] has that same length.



This design leads to algorithm 1, which has the clicking artifacts.
And our new idea was to have a shorter h[n] to prevent inter-window artifacts.

Filter Design

How can we design a filter with 2048 points in time (n) but 8192 points in frequency (k)?

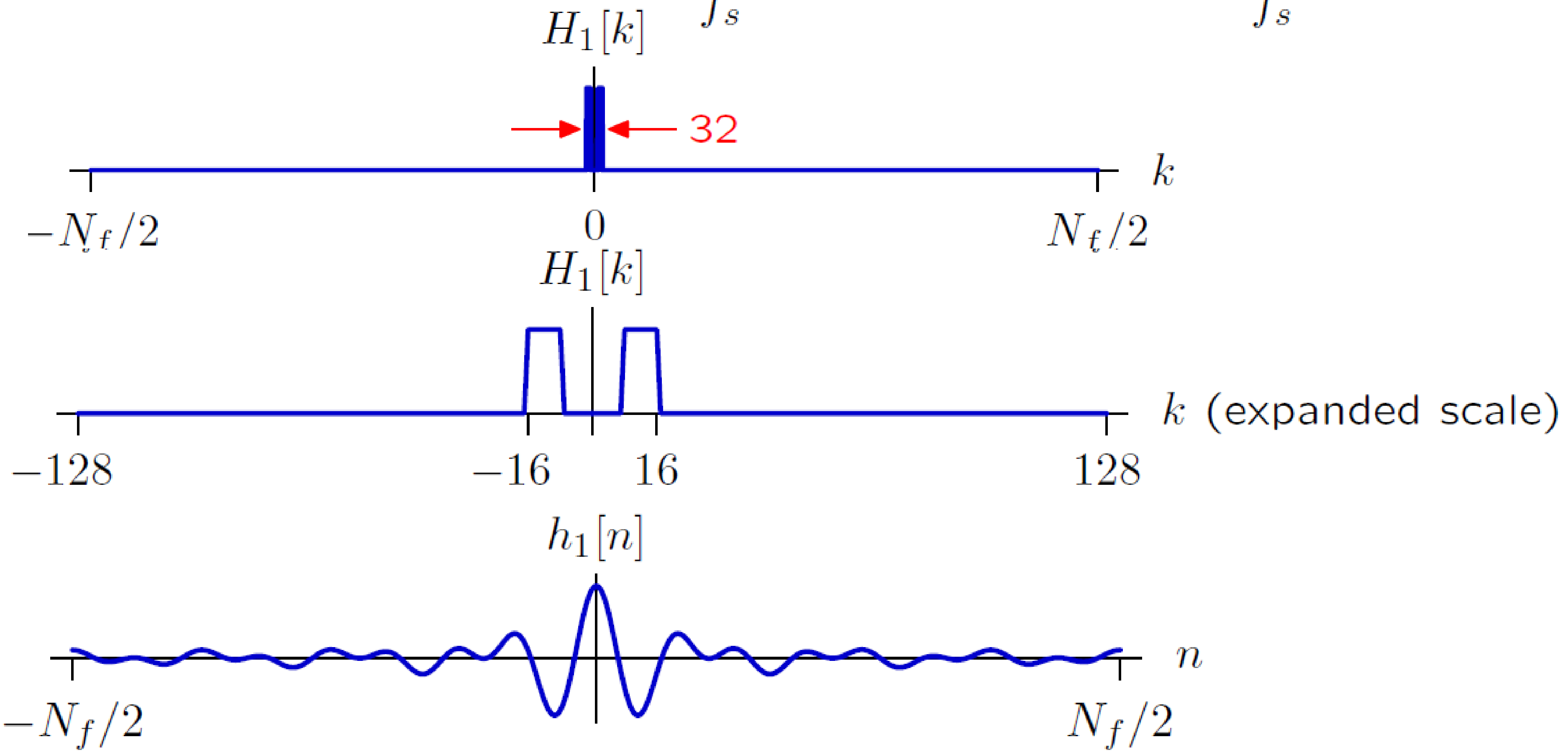
- Start by designing a filter $H_1[k]$ with length $N_f = 2048$. The filter should only pass frequencies in the range $f_l \leq f \leq f_h$.
- Convert $H_1[k]$ to the time domain using an inverse DFT. The length of the resulting $h_1[n]$ will be $N_f = 2048$.
- Define a new filter $h_2[n]$ which is a version of $h_1[n]$ that is zero-padded to a new length of $N = 8192$.
- Convert $h_2[n]$ to the frequency domain to get $H_2[k]$.

The filter $H_2[k]$ will have 8192 values of k but its time-domain representation $h_2[n]$ will have just 2048 non-zero values.

Filter Design

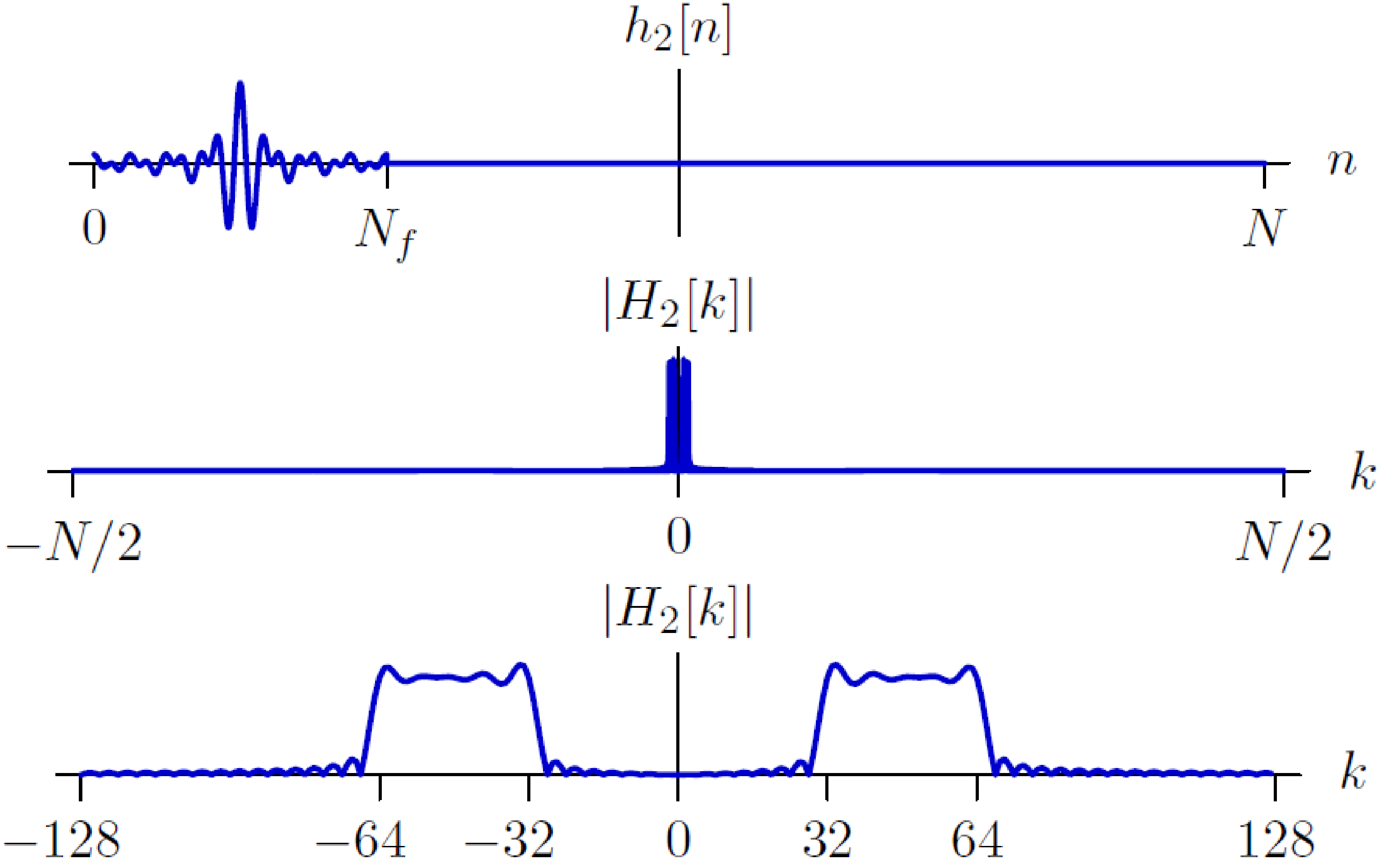
Design a bandpass filter to extract 170-340 Hz frequency region from signal sampled with fs = 44, 100 Hz with N_f = 2048.

$$\frac{170}{f_s} \times N_f \approx 8 \quad \leq \quad k \quad \leq \quad \frac{340}{f_s} \times N_f \approx 16$$



Filter Design

Zero-pad to make filter length equal to window length N = 8192.



Bass and harmony are faintly audible - probably because of deviations from ideal filters. Ripples are due to Gibb's phenomenon.

Listen to result: am_filtered.wav

Significant improvement. No clicks.

Gibb's Phenomenon

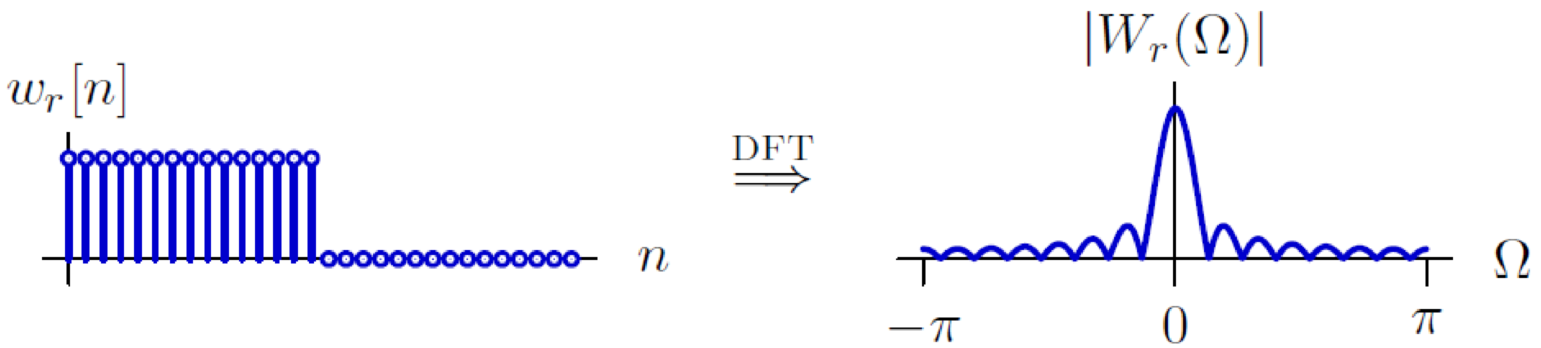
Ripples in frequency result from windowing in time.

A rectangular window in time

$$w_r[n] = \begin{cases} 1 & \text{if } 0 \leq n < N \\ 0 & \text{otherwise} \end{cases}$$

corresponds to a DT sinc in frequency.

$$W_r(\Omega) = \sum_{n=-\infty}^{\infty} w[n]e^{-j\Omega n} = \sum_{n=0}^{N-1} e^{-j\Omega n} = \frac{1 - e^{-j\Omega N}}{1 - e^{-j\Omega}} = \frac{\sin \frac{\Omega N}{2}}{\sin \frac{\Omega}{2}} e^{-j\Omega \frac{(N-1)}{2}}$$



Multiplying the unit sample response $h[n]$ by a window function, convolves the desired bandpass shape with the DT sinc - generating ripples.

Gibb's Phenomenon

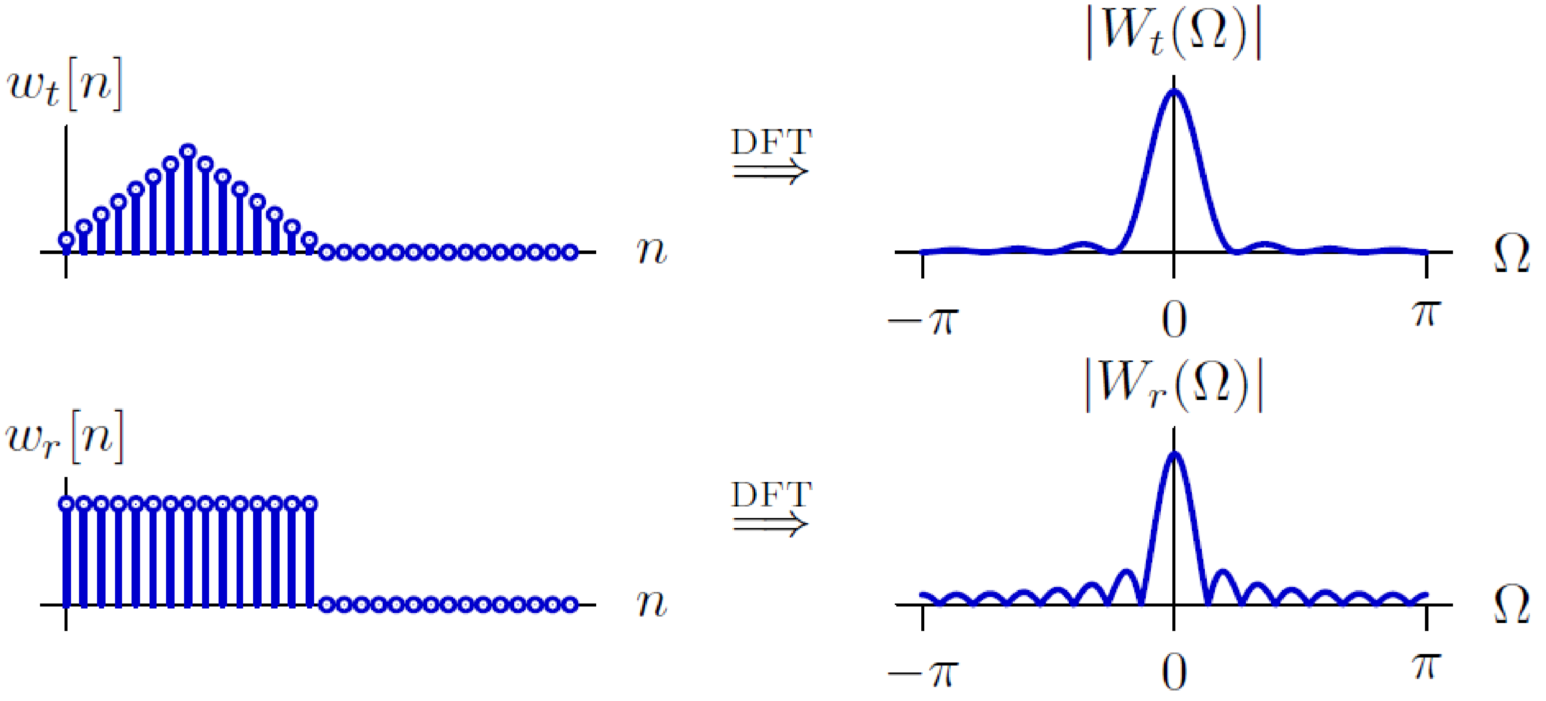
Triangular windows in time produce smaller ripples in frequency.

A triangular window in time

$$w_t[n] = w_r[n] * w_r[n]$$

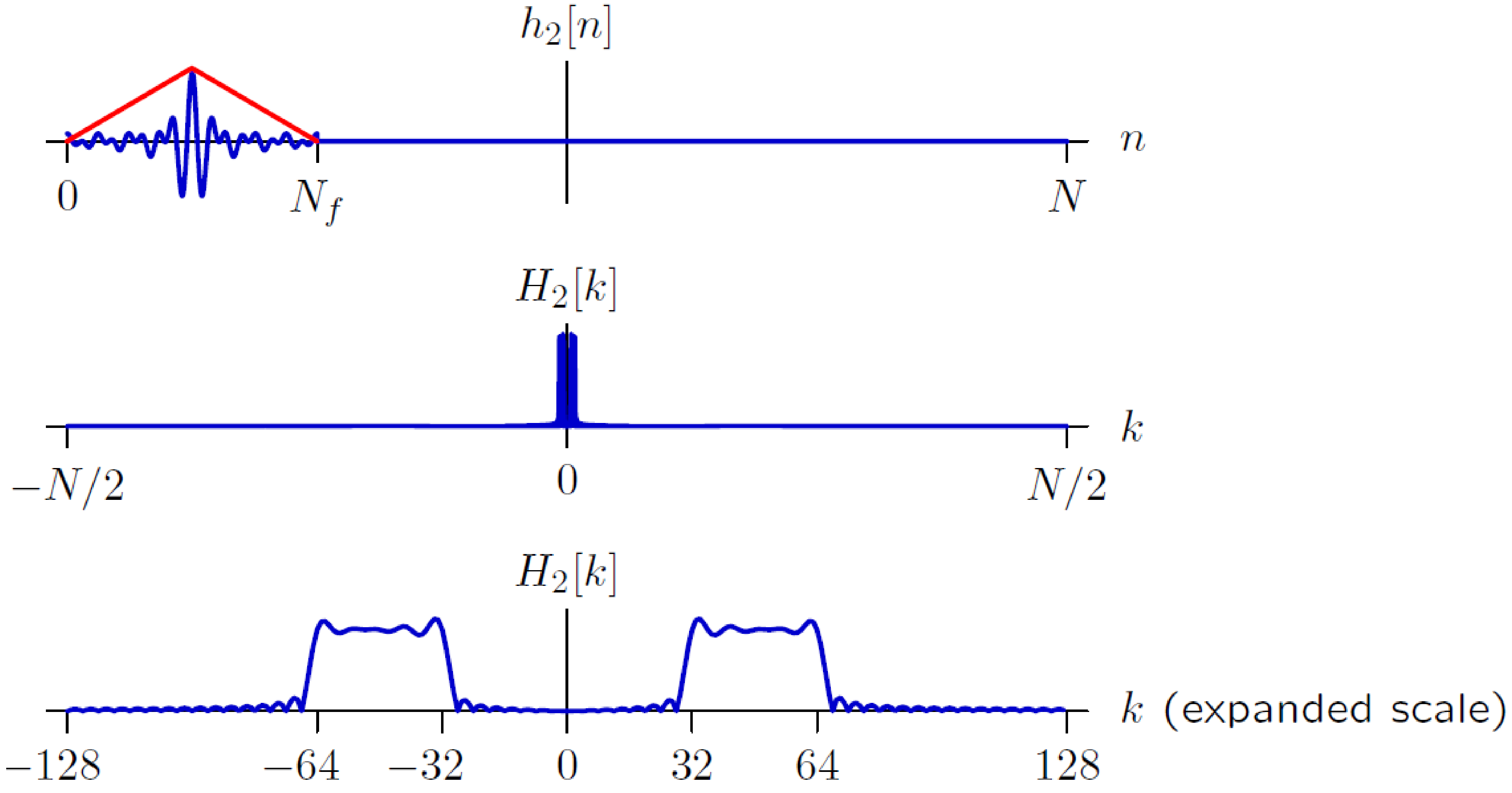
corresponds to a DT sinc squared in frequency.

$$W_t(\Omega) = W_r^2(\Omega)$$



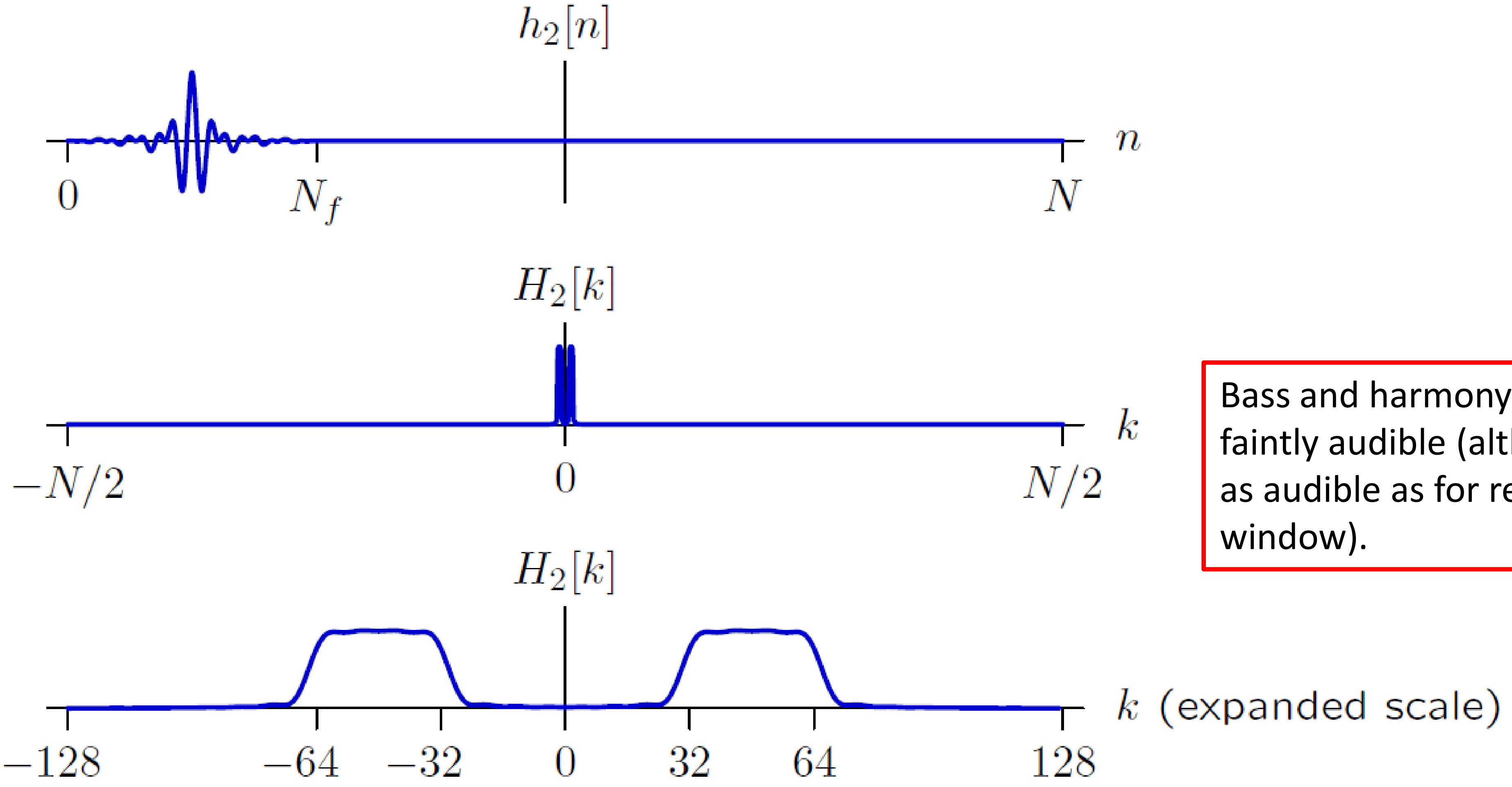
Filter Design

We can reduce the passband ripple by applying a triangular window (red).



Filter Design

We can reduce the passband ripple by applying a triangular window (red).



Bass and harmony are still faintly audible (although not as audible as for rectangular window).

$H2[k]$ is now a smoother function of k, rippling is greatly reduced.

Listen to result: am_triangular.wav

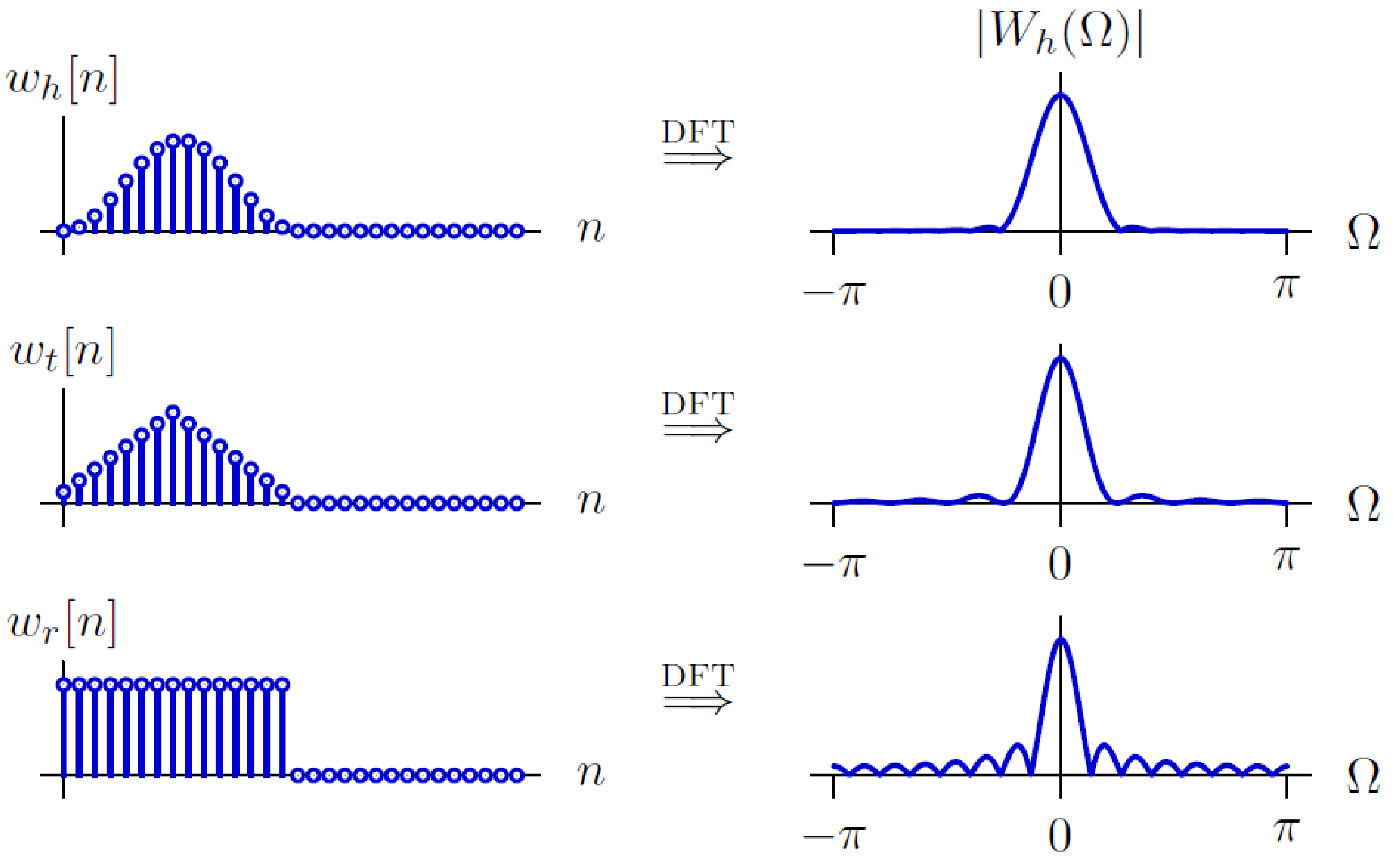
Gibb's Phenomenon

Ripples in frequency result from windowing in time.

A Hann window in time

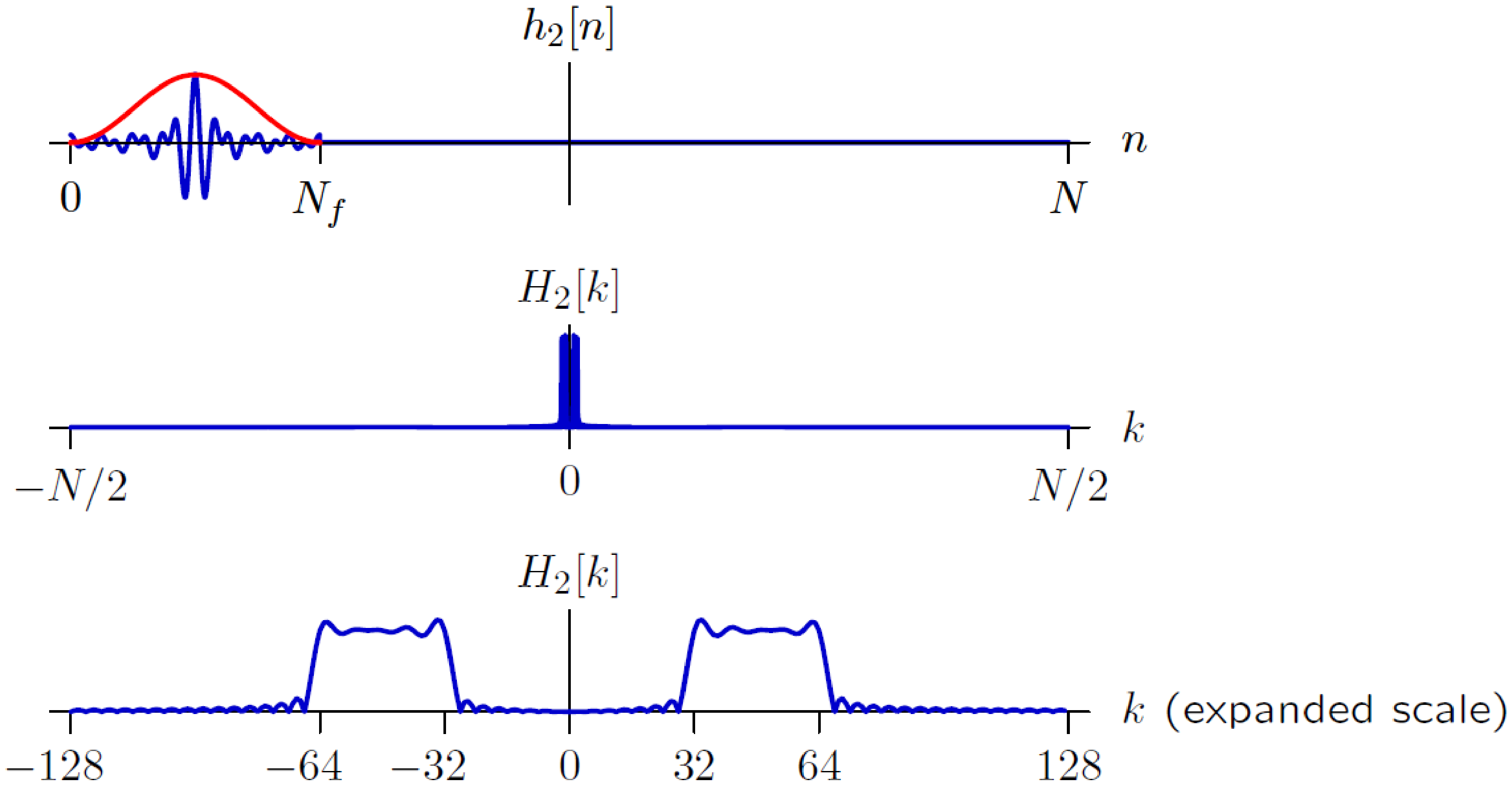
$$w_h[n] = \sin\left(\frac{\pi n}{N}\right)^2$$

produces even smaller ripples.



Filter Design

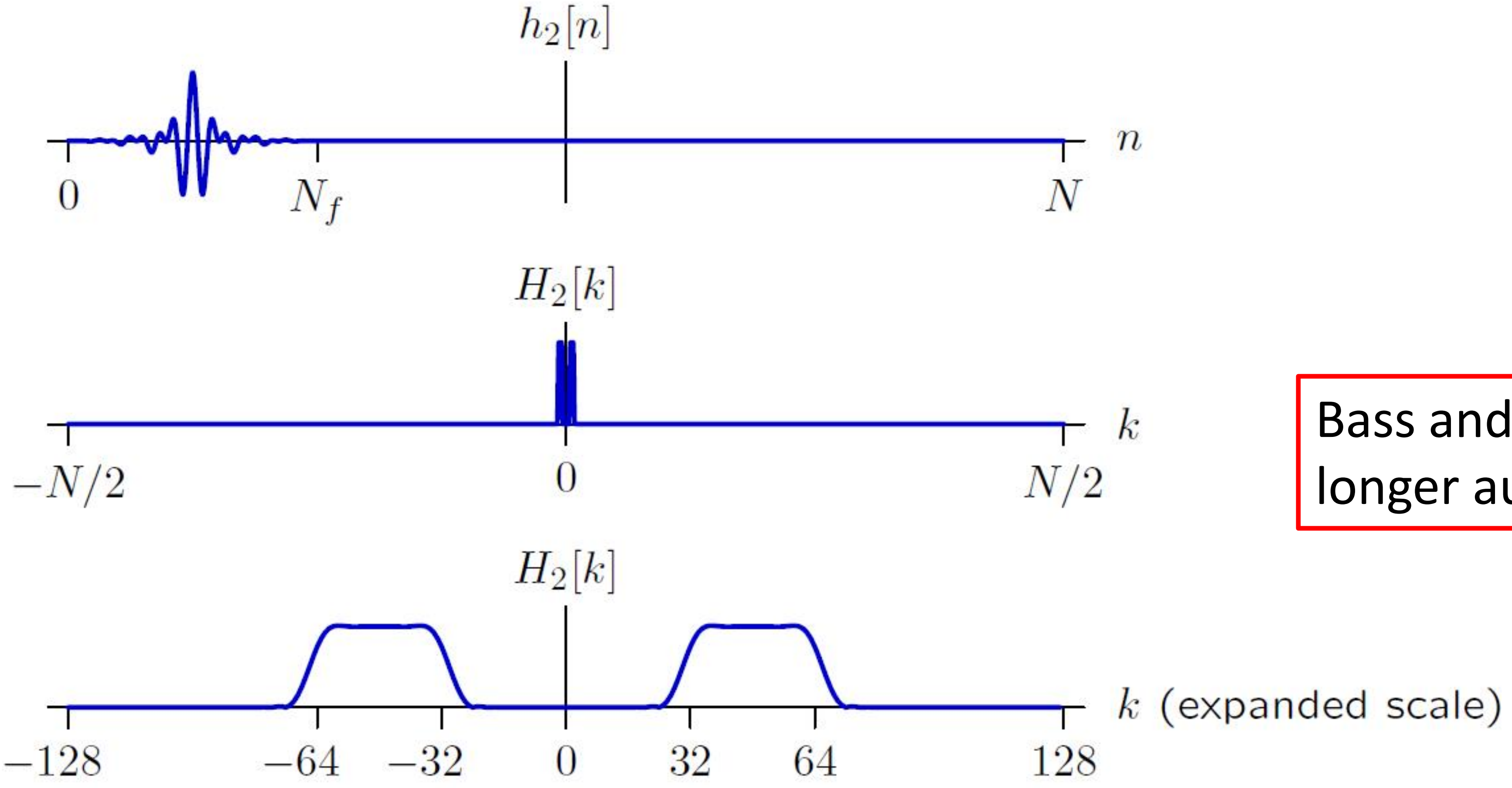
Better yet, try a Hann window (red).



ırmony are still
ble (although not
ıs for rectangular

Filter Design

Better yet, try a Hann window (red).



Bass and harmony no longer audible.

H2[k] is now even smoother.

Listen to result: am_Hann.wav

Summary

Today we learned short-time Fourier Transform. A key feature of the DFT is that it can be applied to parts of a signal.

- to visualize the change in frequency with time, and/or
- to process long signals, one chunk at a time.

We will now go to 4-370 for recitation & common hour