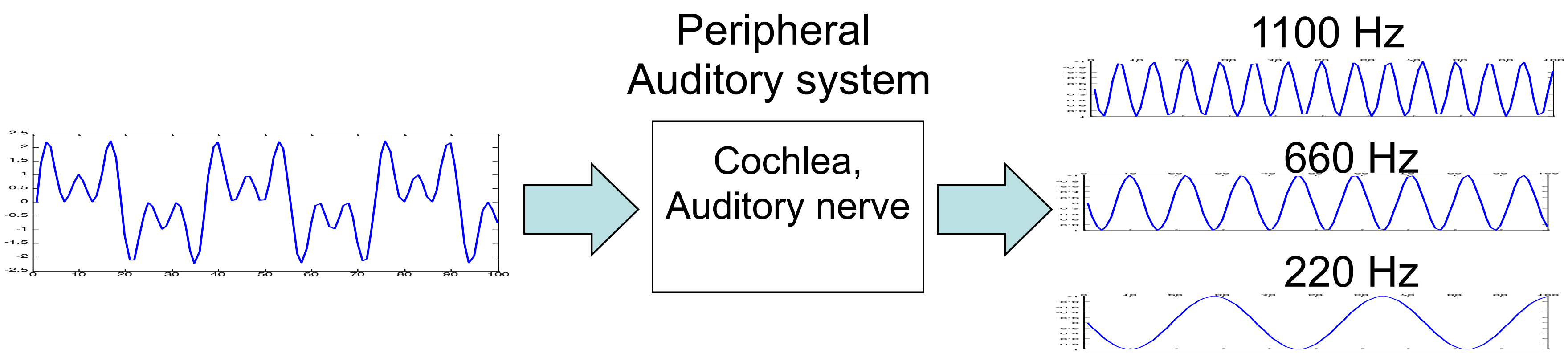
Topic 6

The Digital Fourier Transform

Why bother?

- The ear processes sound by decomposing it into sine waves at different frequencies.
- An algorithm that does the same would be a step towards machines that hears things as humans do.
- So…how do we do this by machine?



Jean Baptiste Joseph Fourier

He was a French mathematician and physicist who lived from 1768-1830.

He presented a paper in 1807 to the Institut de France claiming any continuous signal with finite period could be represented as the sum of an infinite series of properly chosen sinusoidal wave functions at different frequencies.

We now call this the Fourier Series.

Could this be the way to decompose sounds into different frequencies, like the ear does?

Factoid

Among the reviewers of Fourier's paper were two famous mathematicians

- Joseph Louis Lagrange (1736-1813)
- Pierre Simon de Laplace (1749-1827)

Lagrange said sine waves could not perfectly represent signals with discontinuous slopes, like square waves. (He was, technically, right)

Thus, the Institut de France did not publish Fourier's work until 15 years later, after Lagrange died.

The Fourier Series

Given a periodic function x(t)

$$x(t+mT)=x(t)\ \ \forall m \in Integers$$

the period

An **INFINITE** series of sine and cosine functions reproduces x(t)

$$x(t)=A_0+\sum_{n=1}^{\infty}\left[A_n\cos(\omega_n t)+B_n\sin(\omega_n t)\right]$$

Where frequency ω_n is an integer multiple n of fundamental frequency ω_0

$$\omega_n = n\omega_0 = 2\pi n/T$$

Aside: Two defs for "frequency"

- The frequency of a periodic function can be defined in two ways.

$$f = \omega / 2\pi = 1/T$$

Frequency

Angular Frequency

Period

Fourier Series

The frequency of
this component

Quite a few!

$$x(t) = A_0 + \sum_{n=1}^{\infty} \left[A_n \cos(\omega_n t) + B_n \sin(\omega_n t) \right]$$

The original signal

Amplitude: the contribution
of this component

PROBLEM 1

The Fourier Series has an infinite number of sinusoids in it.

It isn't practical to calculate an infinite number of things, in the general case.

We need to frame the problem as a finite one, so we can actually solve the general case.

PROBLEM 2

A function representable by a Fourier series is perfectly periodic. Therefore, it goes on infinitely.

Real world audio is not infinite in length…and things are only locally periodic.

Solution: Sample, Window, Hope

STEP 1: Take a “snapshot” of the signal by sampling it a finite number of times within a brief time window

STEP 2: Pretend what you saw in that window goes on forever

Voila! Now you have a “periodic” signal represented by a finite number of points.

Discrete Fourier Transform

- Represents a **finite** sequence of complex values as a **finite** number of discrete sinusoidal functions.
- This finite sequence of samples can be perfectly reproduced by the finite set of sinusoids.

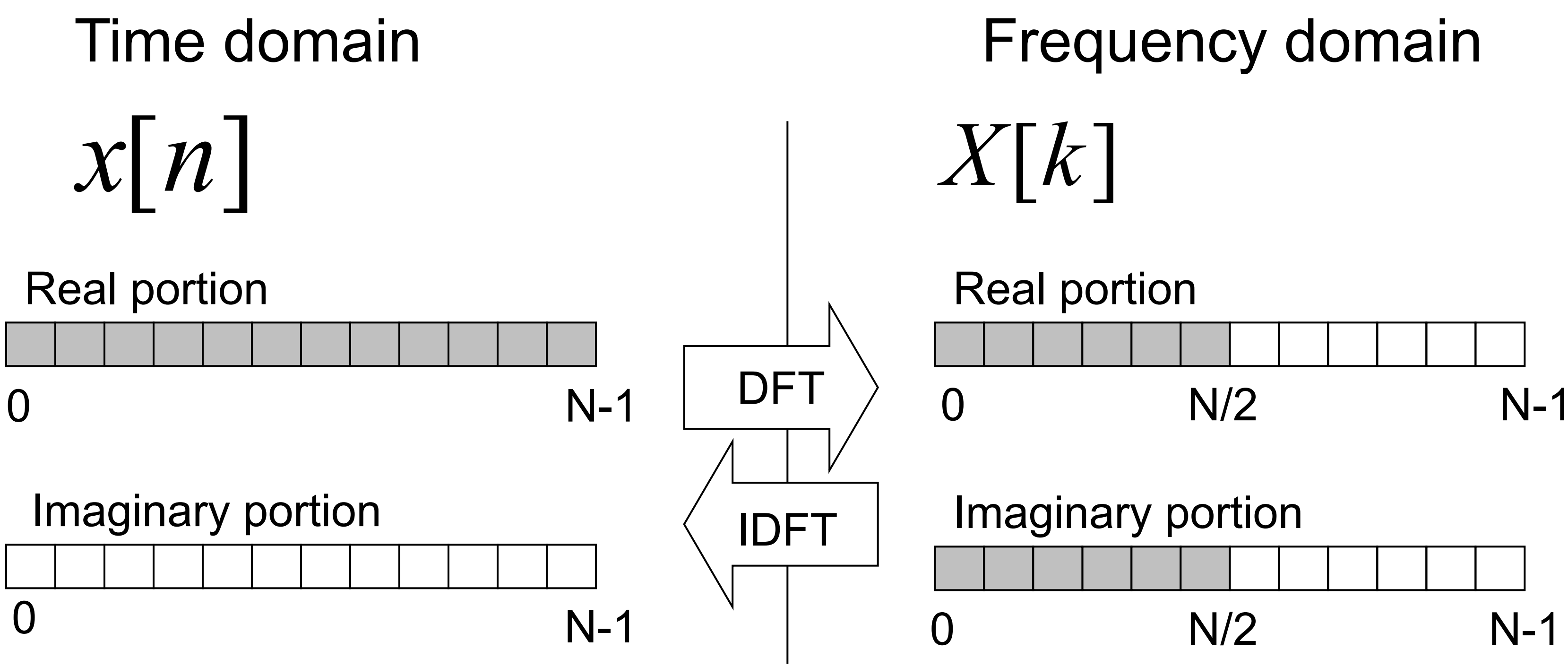
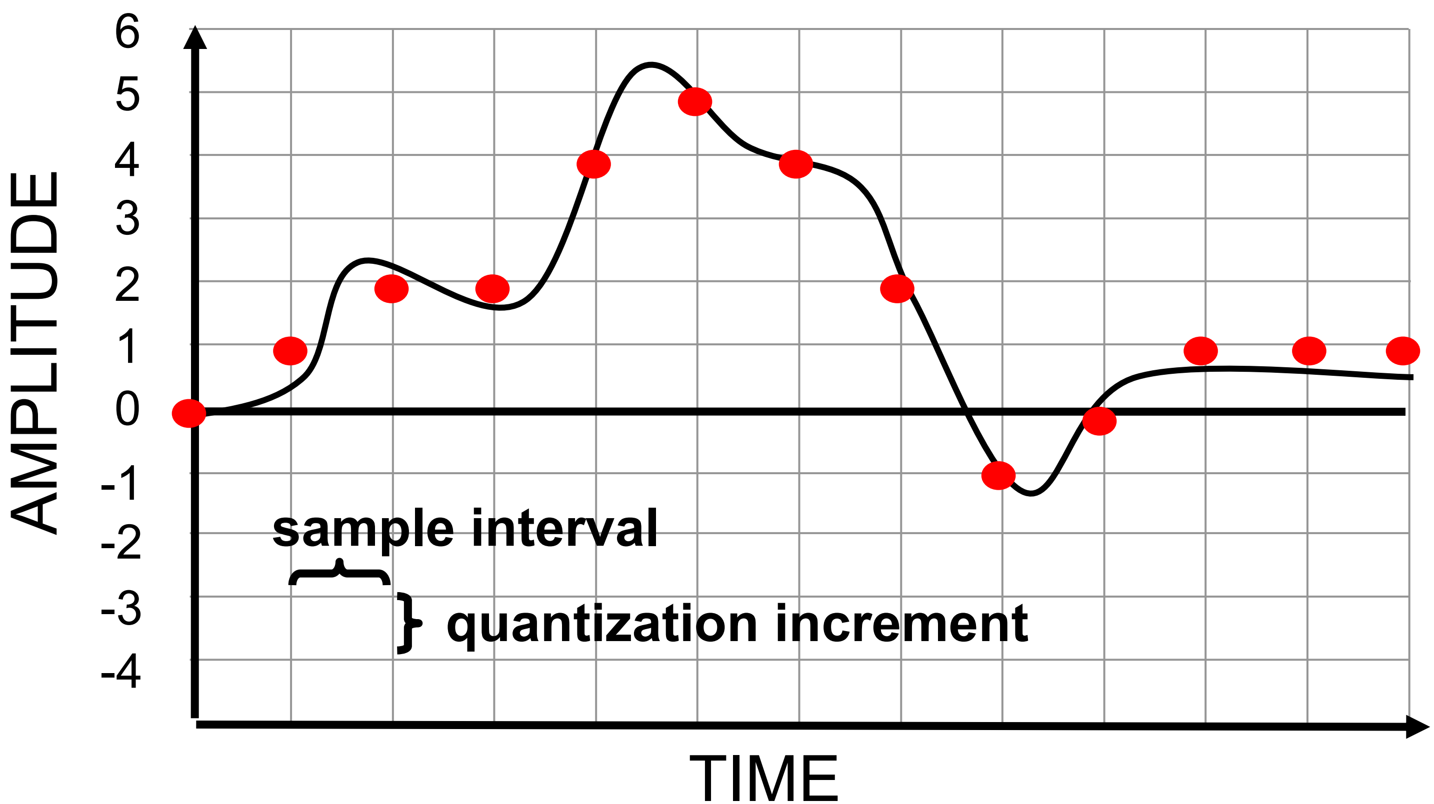


Image from *The Scientist and Engineer's Guide to Digital Signal Processing* by Steven Smith)

Digital Sampling

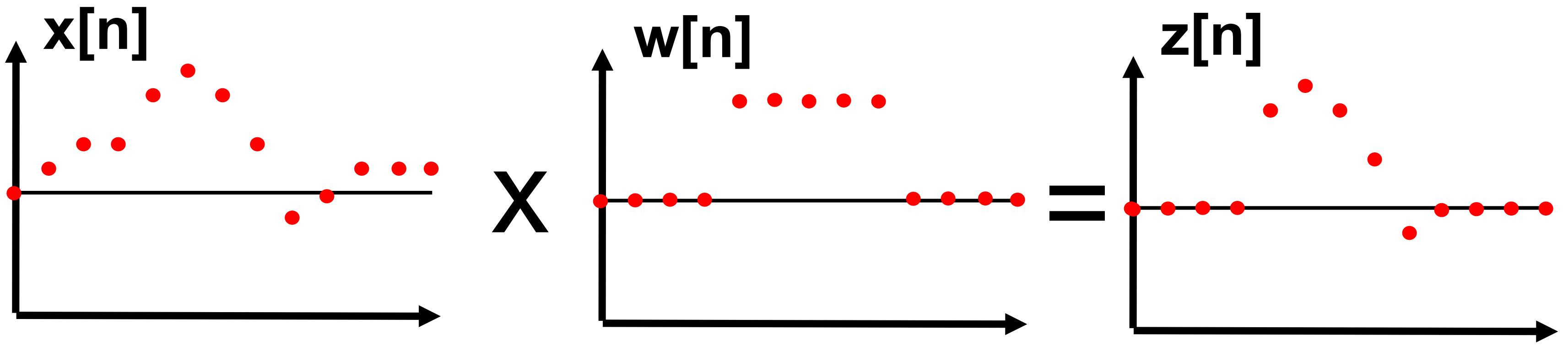
An analog signal is ***sampled*** into sequence of discrete sample points, x[n]



Windowing

x[n] is ***windowed*** by function **w[n]**

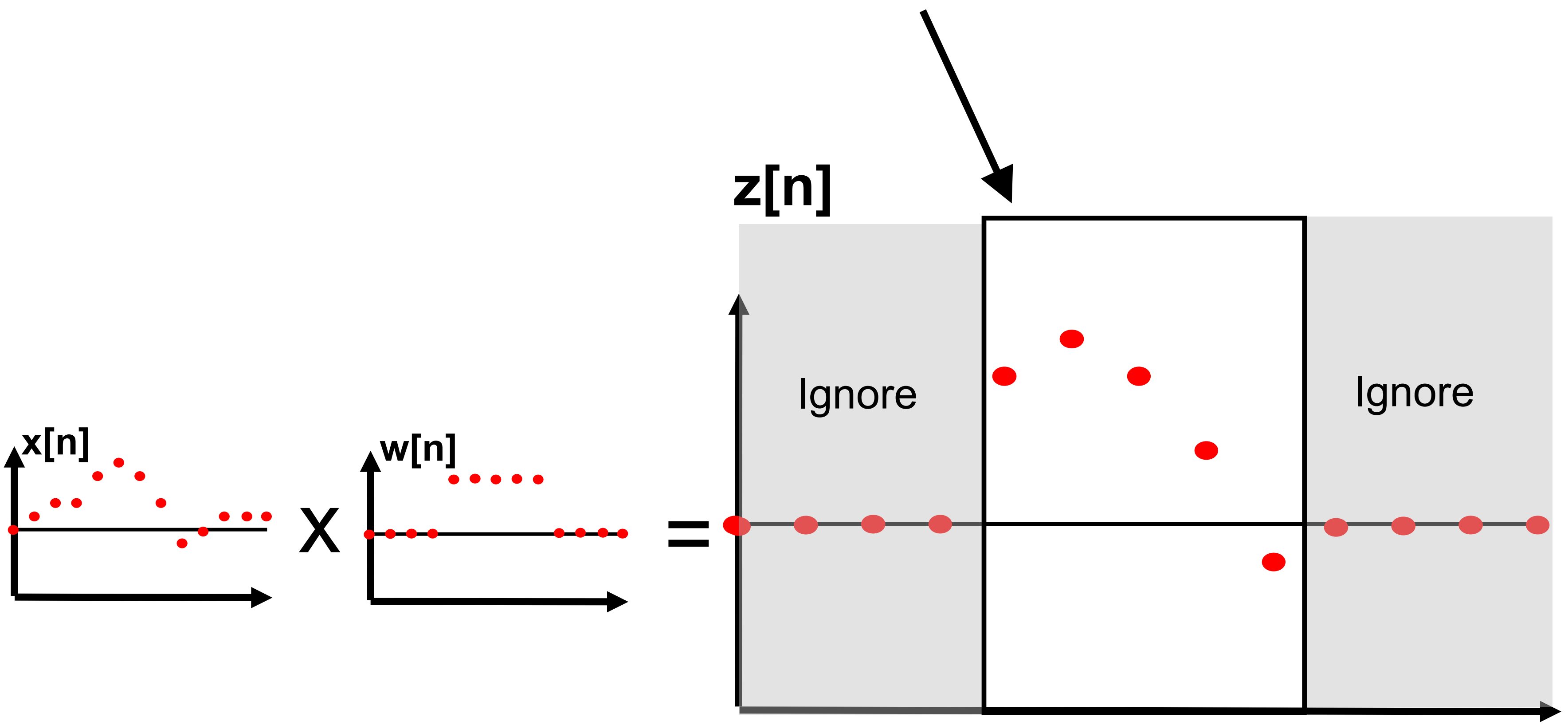
(multiply the *ith* value of **x** by the *ith* value of **w**)



Example: windowing x[n] with a rectangular window

Only what's in the window

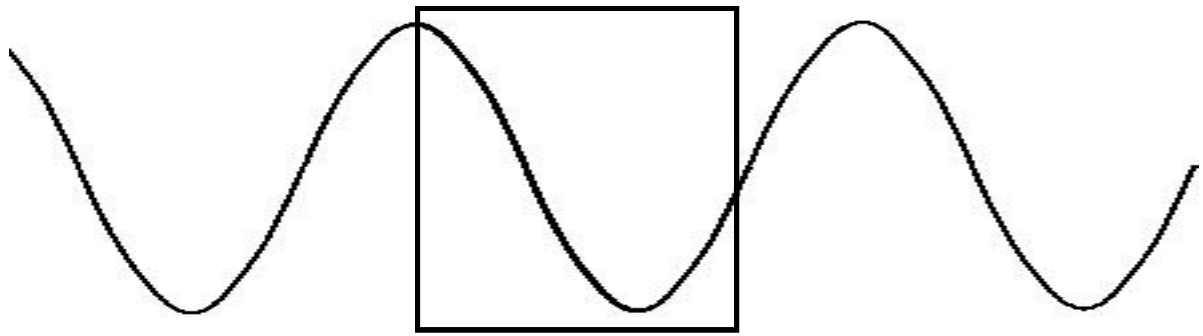
Do the DFT only on the values in the window

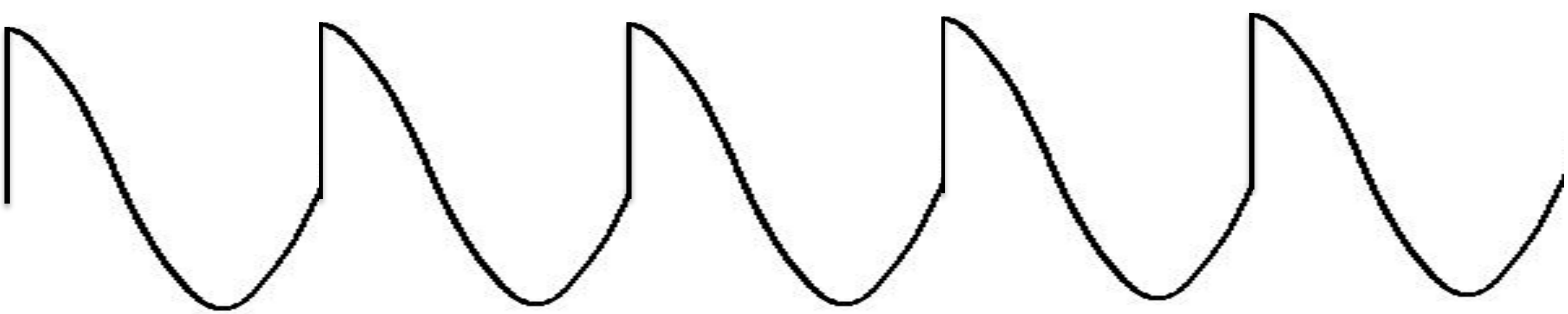


Windowing can lead to problems

The original signal

The sample window





What the Fourier Transform “imagines” the signal looks like, based on what was in the window

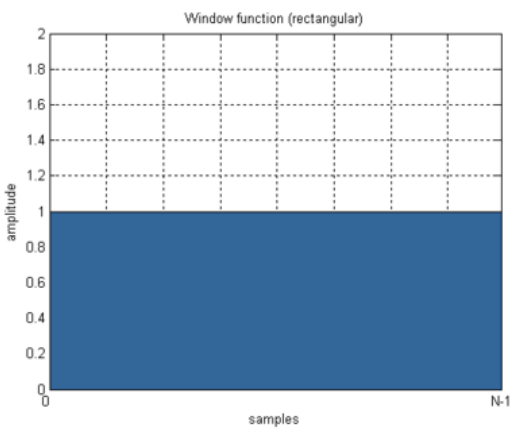
Window size

- Your window should be longer than the period of the function you want to analyze
- At a sample rate of 8000 Hz, what is the minimum window size that can capture the lowest sound you can hear?

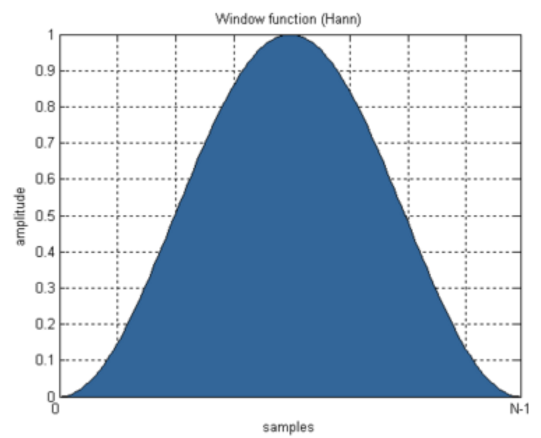
Window shape

- Making the window “small” at the edges reduces weirdness.

BAD



GOOD



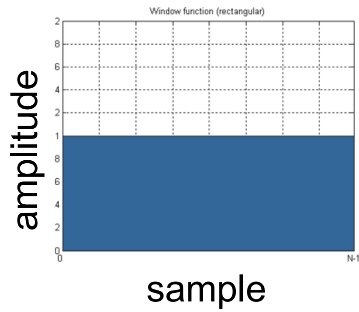
Why window shape matters

- Don't forget that a DFT assumes the signal in the window is periodic
- The boundary conditions mess things up…unless you manage to have a window whose length is an exact integer multiple of the period of your signal
- Making the edges of the window less prominent helps suppress undesirable artifacts

Some famous windows

- Rectangular

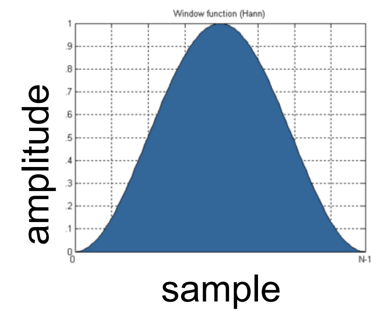
$$w[n] = 1$$



Note: we assume w[*n*] = 0 outside some range [0,*N*]

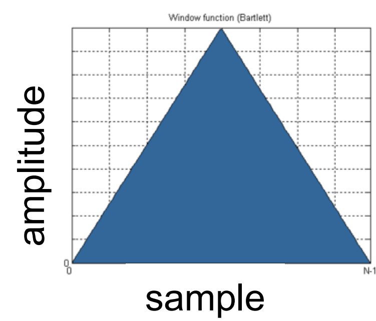
- Hann (Julius von Hann)

$$w[n] = 0.5\left(1 - \cos\left(\frac{2\pi n}{N-1}\right)\right)$$



- Bartlett

$$w[n] = \frac{2}{N-1}\left(\frac{N-1}{2} - \left|n - \frac{N-1}{2}\right|\right)$$



The DFT and IDFT formulae

What's different between them

Discrete Fourier Transform

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{\boxed{-}\frac{2\pi i}{N}kn}$$

Inverse Discrete Fourier Transform

$$x[n] = \boxed{\frac{1}{N}} \sum_{k=0}^{N-1} X[k] e^{\frac{2\pi i}{N}kn}$$

Some points

- If you have n samples in your window, you will get n frequencies of analysis from your FFT.
- If the samples were grabbed at regular times, then the analysis frequencies will be spaced regularly in the frequency domain.

Fundamental Frequency of a Signal

- The lowest frequency a sine or cosine can have and still fit into one period of the function.
- Often called "F zero"
- Don't confuse this for the DC offset.
- Don't confuse the fundamental frequency of a signal with the fundamental frequency of analysis of an FFT

$$f_0 = \omega_0 / 2\pi = 1/T$$

Fundamental Frequency of Analysis

- The lowest frequency component a Fourier transform can analyze meaningfully
- All the frequencies of analysis are integer multiples of the fundamental frequency of analysis
- This is also the spacing between frequencies of analysis.

$$f_{analysis} = \frac{S}{N}$$

Sample rate (S)

Number of samples in my window (N)

About Frequencies of Analysis

- A recording with N points produces a DFT with N points.
- The energy in the DFT is symmetric around two “pivot points”: the DC offset (the 0 frequency) and ½ the sample frequency (S).
- Let’s look at an example.

The numbers in an 8 point FFT

Array Index	0	1	2	3	4	5	6	7
8 point signal	-0.13	-0.98	0.38	-0.27	-0.13	-0.27	-0.63	-0.98
FFT of signal	-3 -i	-1-i	0	1+i	2	1-i	0	-1+i

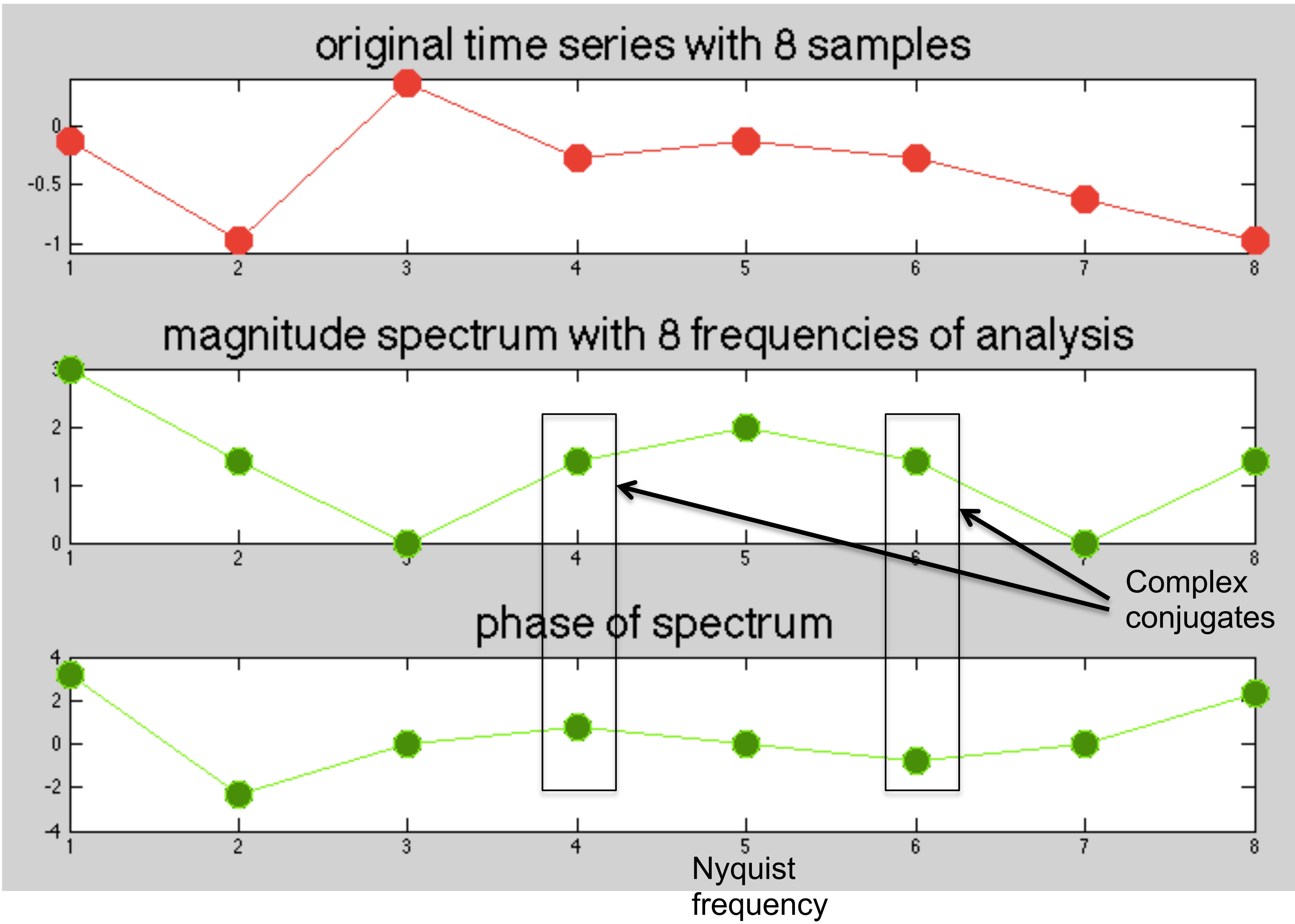
DC offset

Nyquist frequency

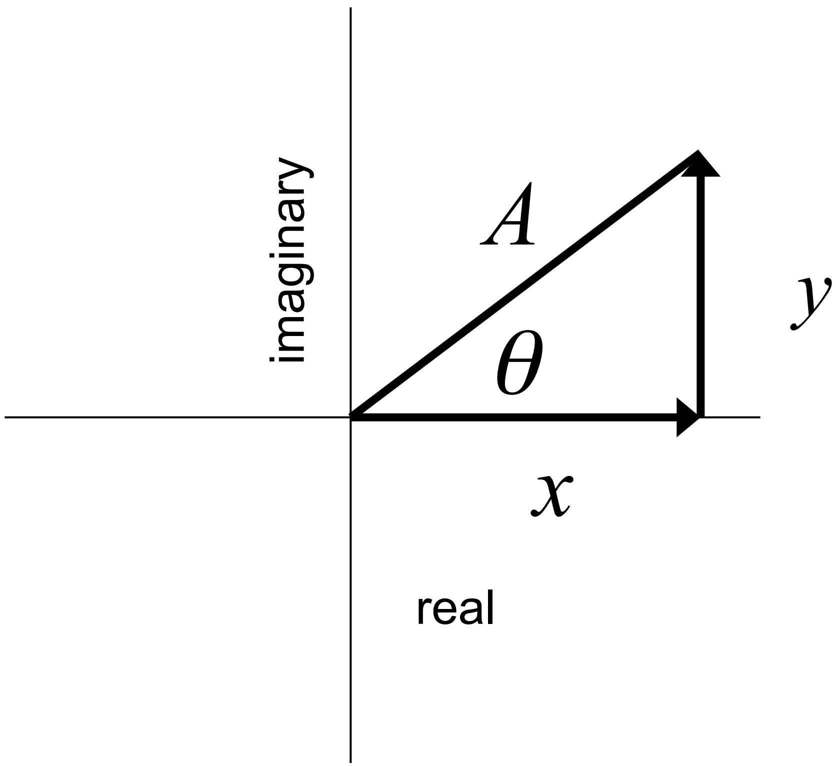
Complex conjugates

Better index	0	1	2	3	4 (-4)	-3	-2	-1
Frequency of FFT	0	S/N	2S/N	3S/N	4S/N	-3S/N	-2S/N	-S/N
If S=1000 Hz and N = 8	0	125Hz	250Hz	375Hz	500Hz	-375Hz	-250Hz	-125Hz

Another view of the same data



Complex Numbers



$$z \equiv x + iy$$

$$\equiv A\left(\cos\theta + i\sin\theta\right)$$

$$x = A\cos\theta$$

$$y = A\sin\theta$$

$$A = \sqrt{\left(x^2 + y^2\right)}$$

Euler's Formula

- Useful for relating polar coordinates to rectangular coordinates

$$e^{i\theta} = \cos\theta + i\sin\theta$$

Thus...

PHASE

$$z = Ae^{i\theta}$$

AMPLITUDE

Multiplying Complex Numbers

- POLAR notation EASIER for this

$$z_1 = A_1 e^{i\theta_1} \qquad z_2 = A_2 e^{i\theta_2}$$

$$z_3 = \left(A_1 A_2\right) e^{i\left(\theta_1 + \theta_2\right)}$$

Multiplying Complex Numbers

- Cartesian works as follows…

$$z_1 = x_1 + iy_1 \qquad z_2 = x_2 + iy_2$$

$$z_3 = x_1 x_2 - y_1 y_2 + i\left(x_1 y_2 + x_2 y_1\right)$$

Complex Conjugate

- the **complex conjugate** of a complex number is given by changing the sign of the imaginary part.

A complex number $$z = a + ib$$

Its complex conjugate $$\bar{z} = a - ib$$

DFT in Cartesian Coordinates

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-\frac{2\pi i}{N} kn}$$

$$= \sum_{n=0}^{N-1} x[n] \left(\cos\left(-\frac{2\pi kn}{N} \right) + i \sin\left(-\frac{2\pi kn}{N} \right) \right)$$

I got this using Euler's formula.

In Cartesian coordinates, the fact that these are complex values is more obvious.

The Inverse DFT

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{\frac{2\pi i}{N} kn}$$

complex number

$$= \frac{1}{N} \sum_{k=0}^{N-1} \boxed{X[k]} \boxed{\left(\cos\left(\frac{2\pi kn}{N} \right) + i \sin\left(\frac{2\pi kn}{N} \right) \right)}$$

Computational complexity

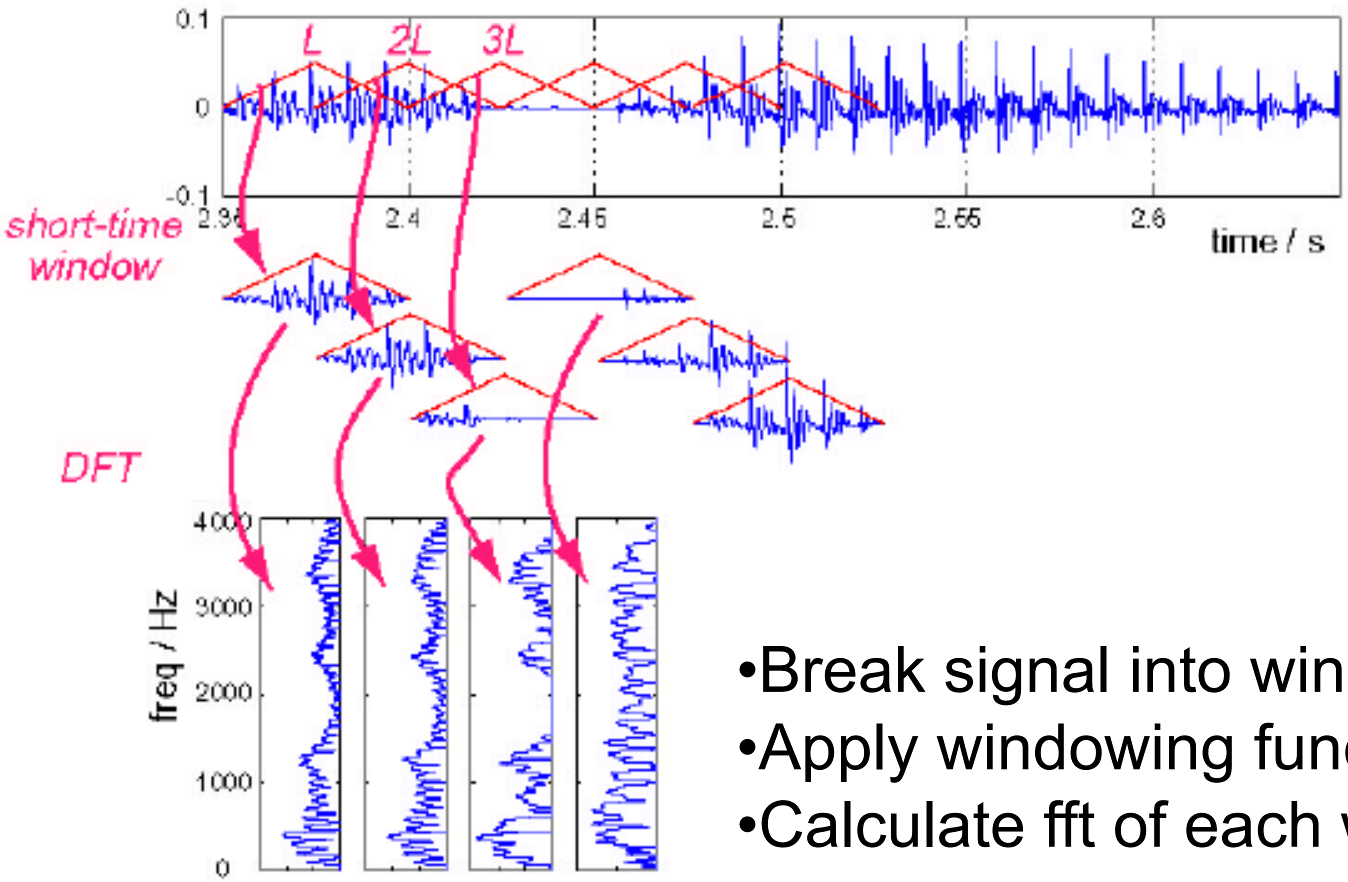
- How many operations does this take for each frequency?
- How many operations total?

$$X[k] = \sum_{n=0}^{N-1} x[n]\left(\cos\left(-\frac{2\pi kn}{N}\right) + i\sin\left(-\frac{2\pi kn}{N}\right)\right)$$

The FFT

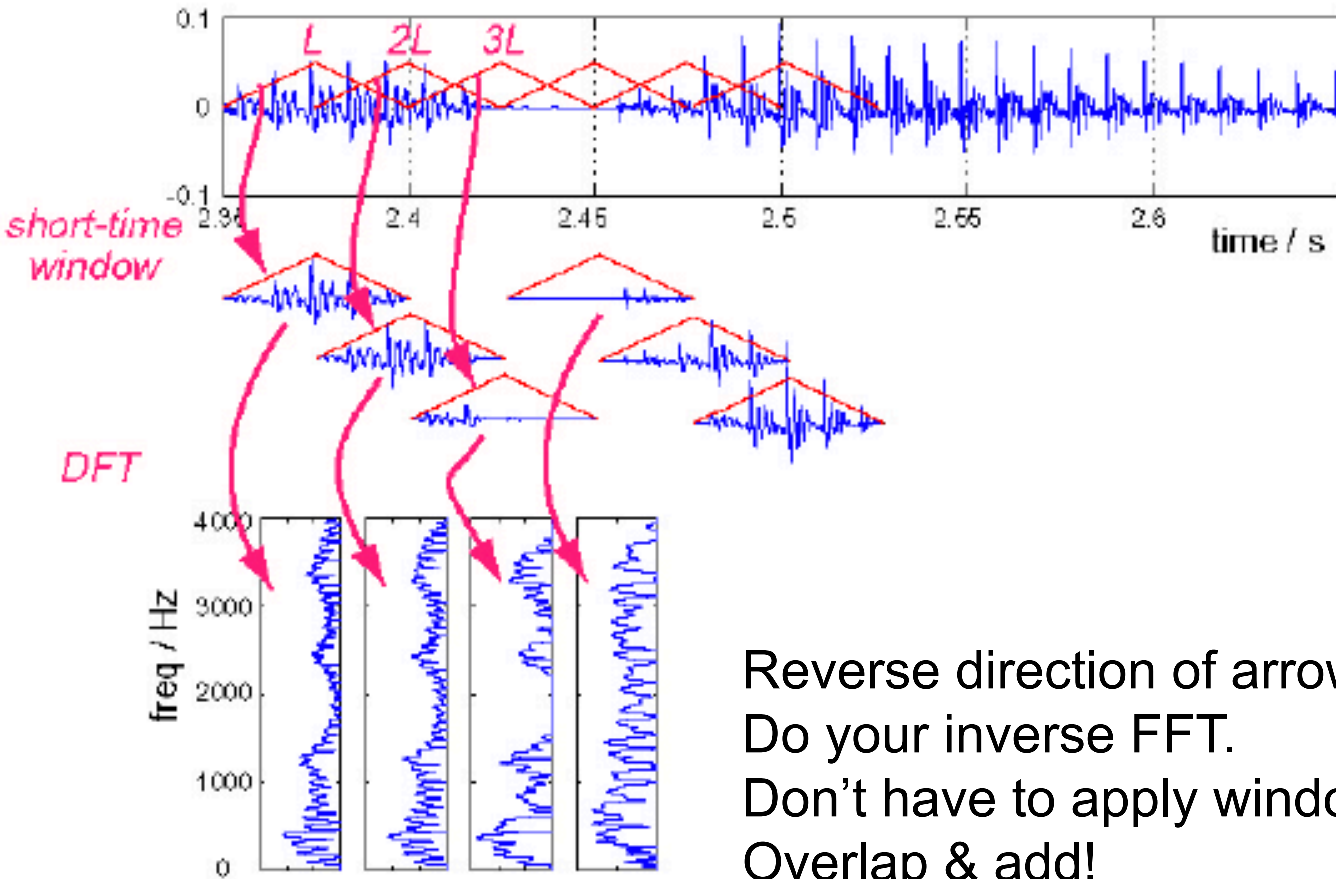
- Fast Fourier Transform
  - A much, much faster way to do the DFT
  - Introduced by Carl F. Gauss in 1805 (ish)
  - Rediscovered by Cooley & Tukey in 1965
  - Big O notation for this is **O(N *log* N)**
  - Matlab functions **fft** and **ifft** are standard
  - **<u>REQUIRES</u>** the window size be a power of 2

Short time Fourier Transform



- •Break signal into windows
- •Apply windowing function
- •Calculate fft of each window

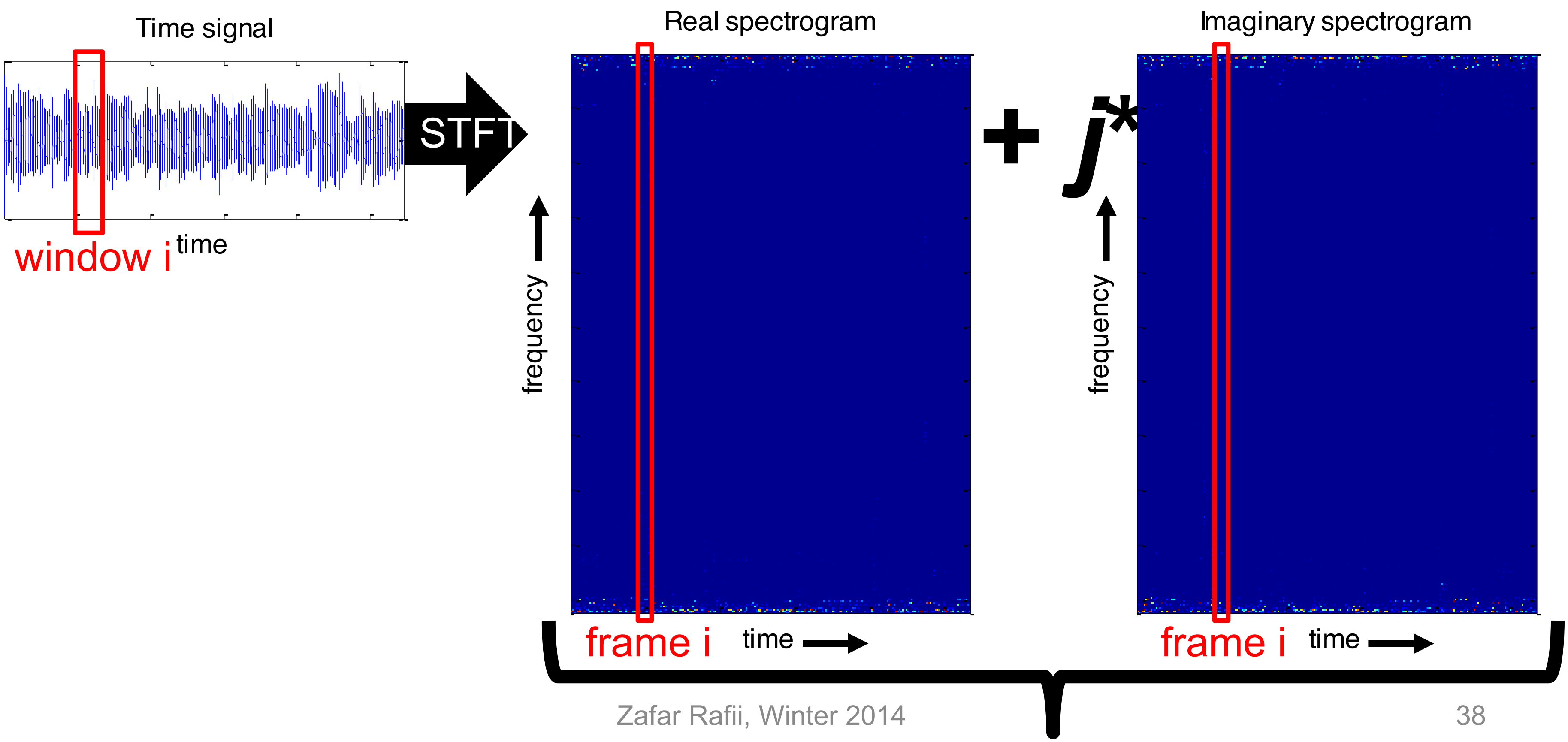
Inverse Short time Fourier Transform



Reverse direction of arrows!
Do your inverse FFT.
Don't have to apply window function again.
Overlap & add!

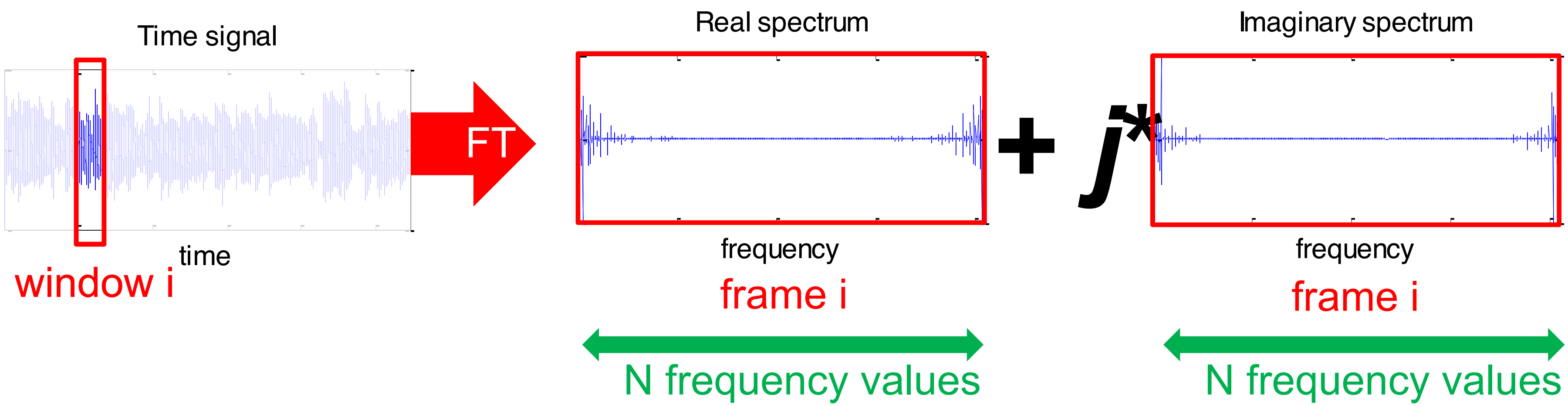
STFT

- The **Short-Time Fourier Transform** (STFT) is a succession of local Fourier Transforms (FT)



STFT

- If we used a window of N samples, the FT has N values, from 0 to N-1; e.g., if N = 8…



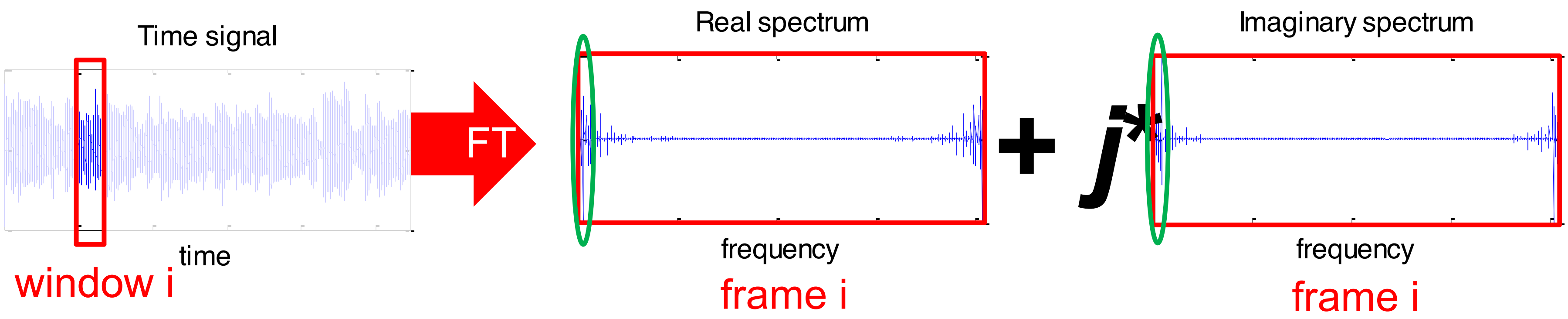
-3	-1	0	1	2	1	0	-1
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*

+ *j**

0	-1	0	1	0	-1	0	1
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*

STFT

- Frequency index 0 is the **DC component**; it is always real. It is an offset from 0. That's all.



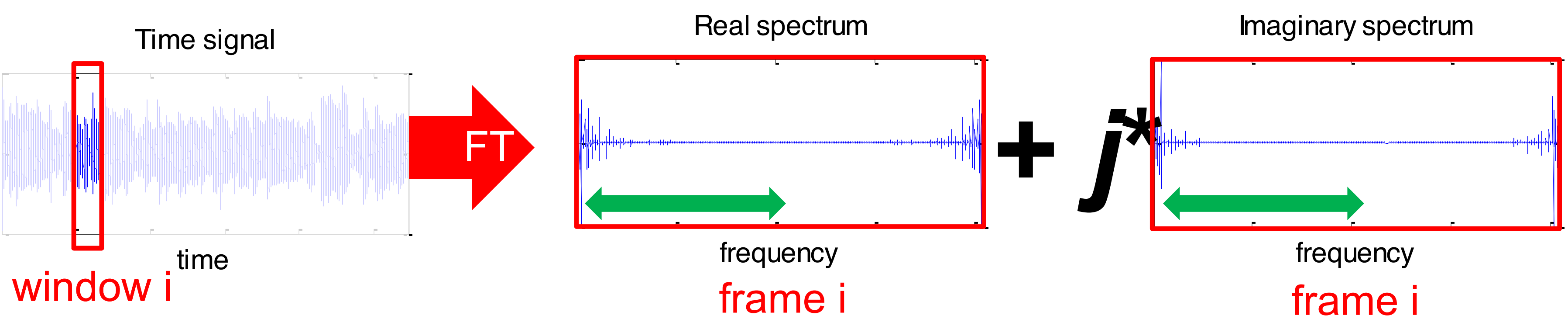
-3	**-1**	**0**	**1**	**2**	**1**	**0**	**-1**
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*

+ *j**

0	**-1**	**0**	**1**	**0**	**-1**	**0**	**1**
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*

STFT

- Frequency indices from 1 to floor(N/2) are the “unique” **complex values (a + *j**b)**



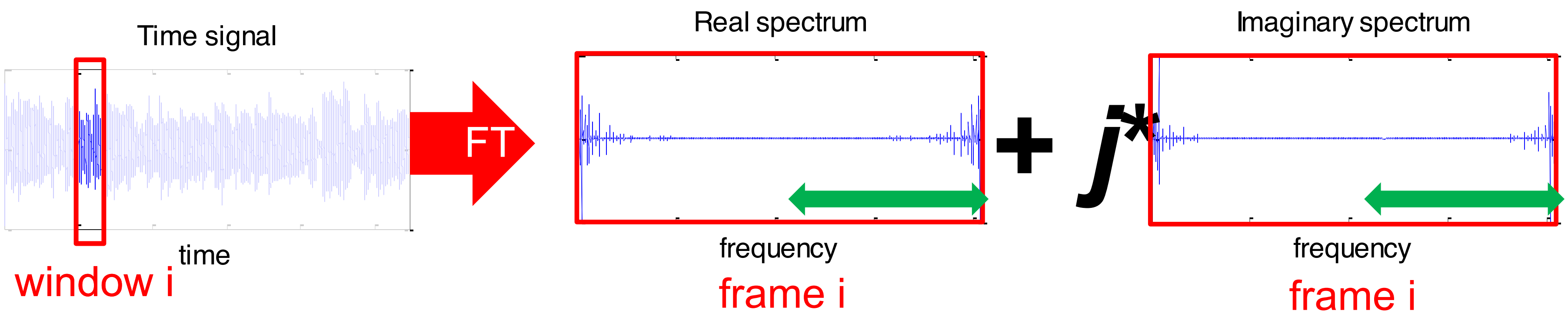
0	1	2	3	4	5	6	7
-3	**-1**	**0**	**1**	**2**	**1**	**0**	**-1**

+ *j**

0	1	2	3	4	5	6	7
0	**-1**	**0**	**1**	**0**	**-1**	**0**	**1**

STFT

- Frequency indices from floor(N/2) to N-1 are the “mirrored” **complex conjugates (a - *j**b)**



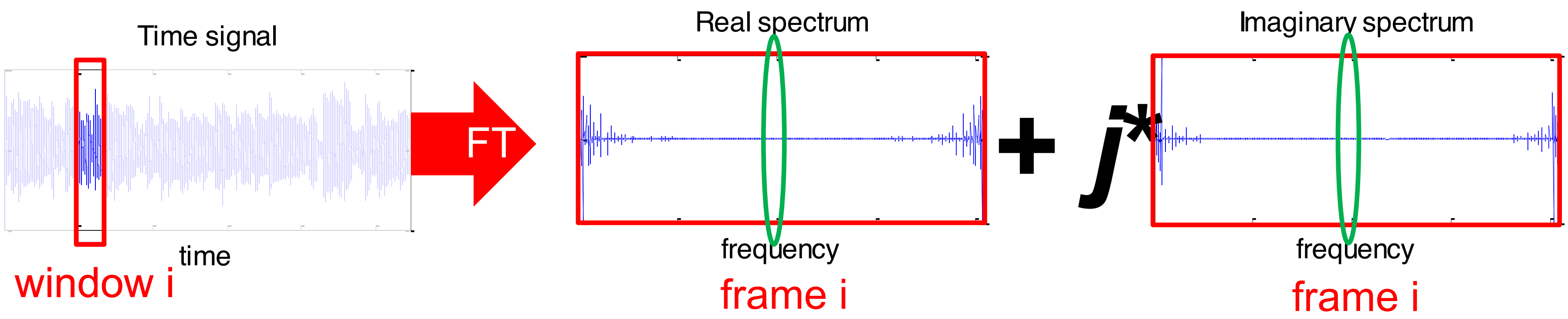
-3	-1	0	1	2	1	0	-1
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*

+ *j**

0	-1	0	1	0	-1	0	1
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*

STFT

- If N is even, there is a **pivot component** at frequency index N/2; it is always real!



-3	-1	0	1	2	1	0	-1
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*

+ *j**

0	-1	0	1	0	-1	0	1
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*

STFT

- Summary of the frequency indices and values in the STFT (in colors!)

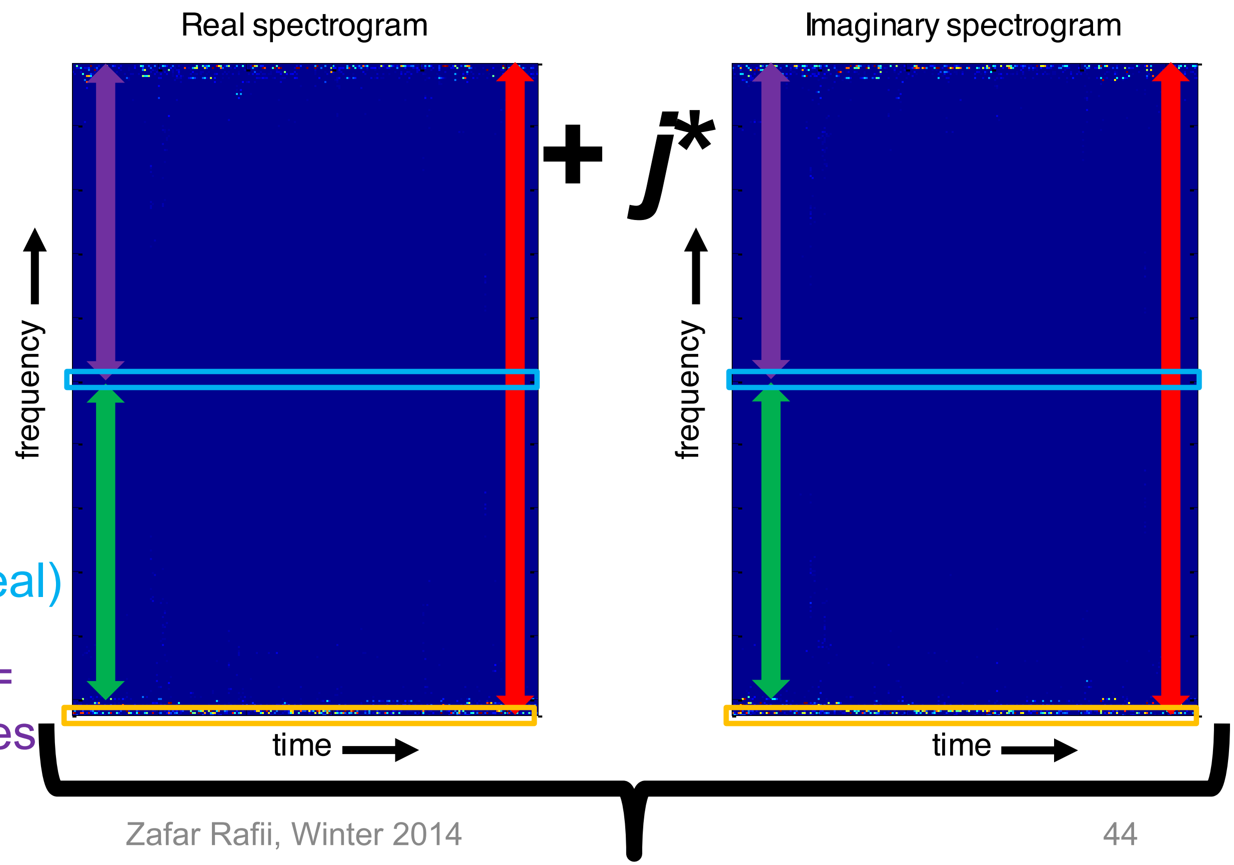
N frequency values =
frequency 0 to N-1

Frequency 0 =
DC component (always real)

Frequency 1 to floor(N/2) =
“unique” complex values

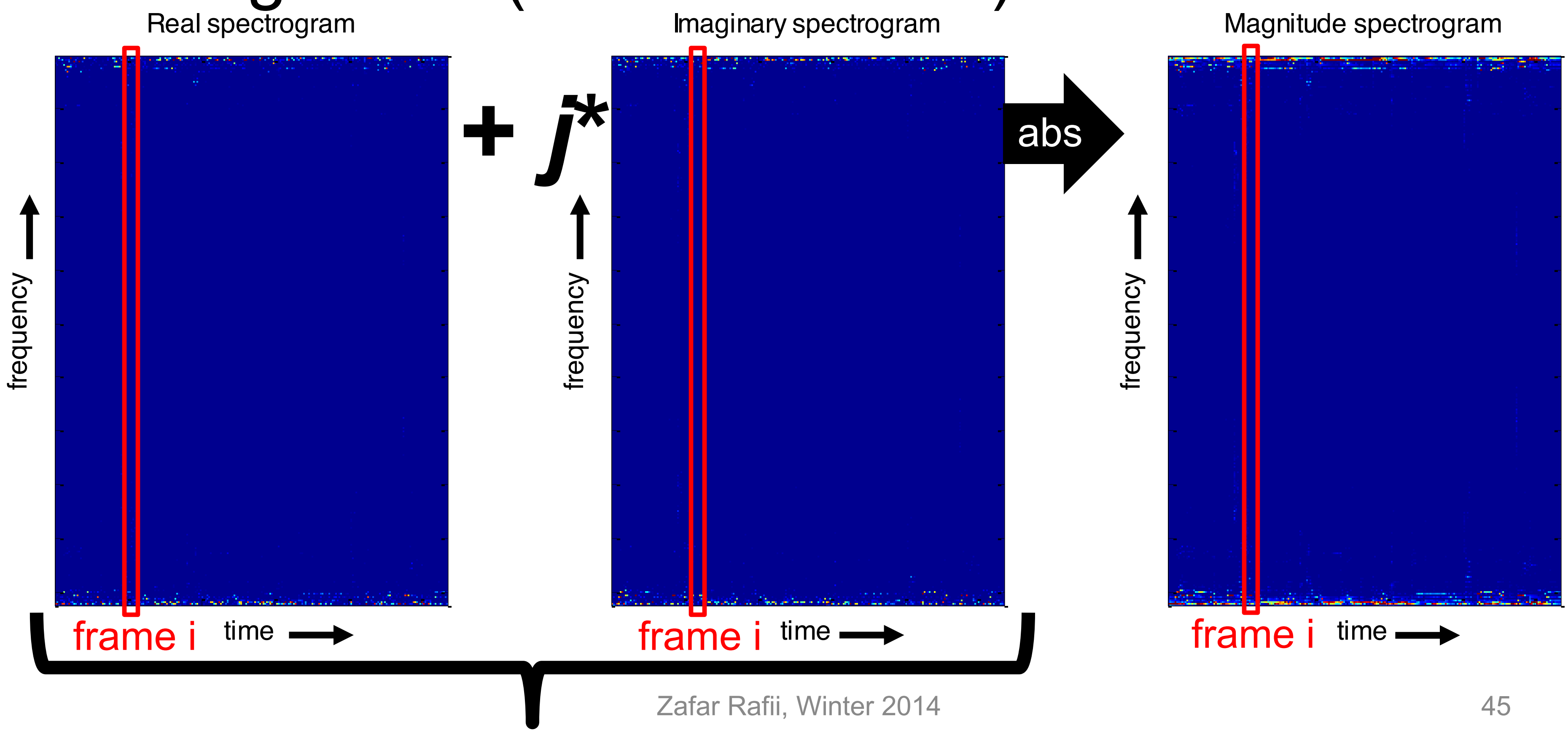
Frequency N/2 =
“pivot” component (always real)

Frequency floor(N/2) to N-1 =
“mirrored” complex conjugates



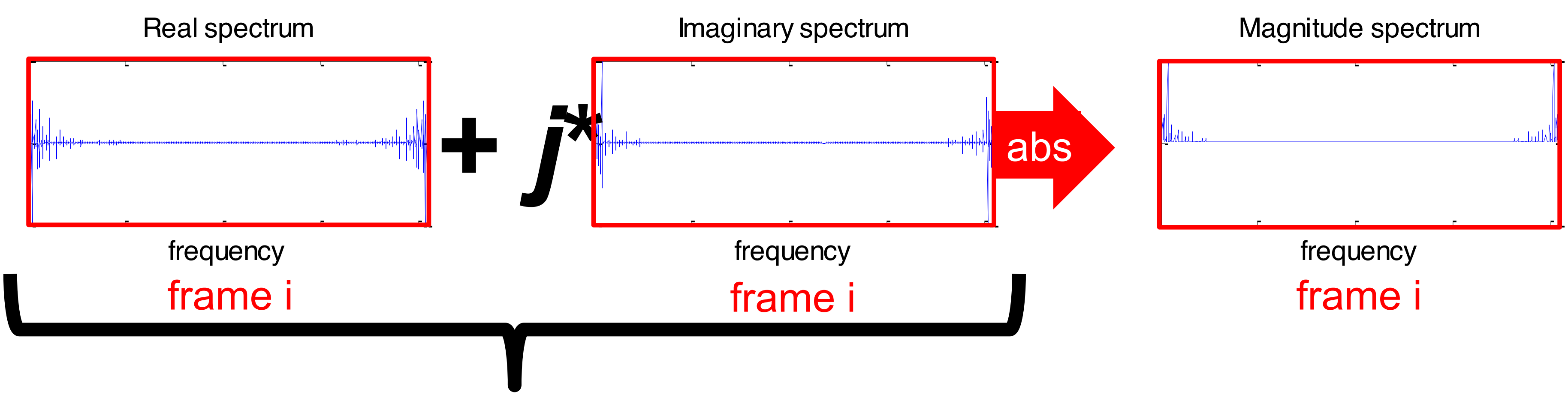
Spectrogram

- The (magnitude) **spectrogram** is the magnitude (absolute value) of the STFT



Spectrogram

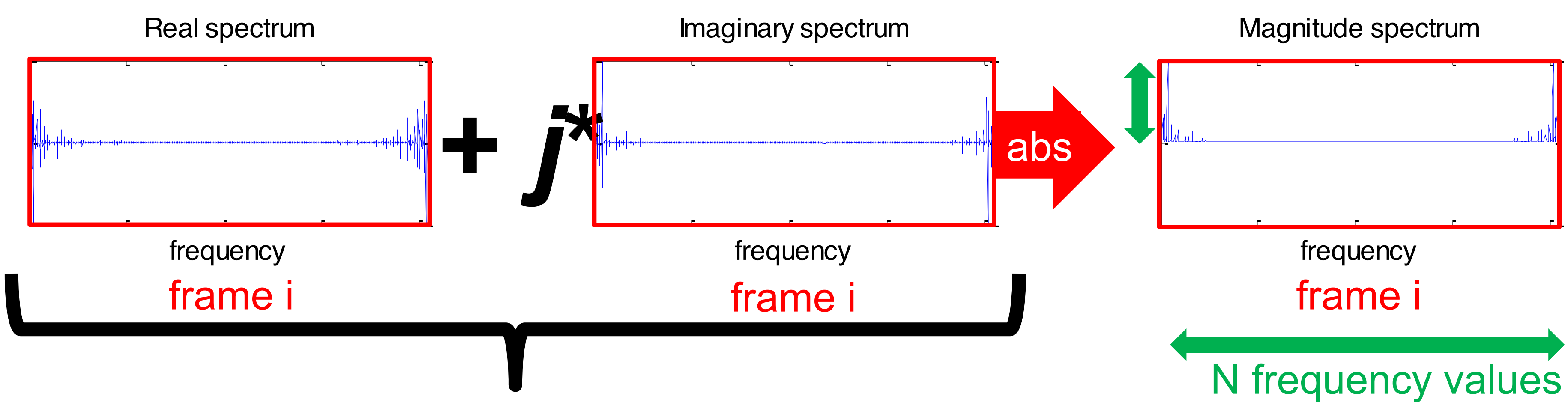
- For a complex number $a + j * b$, the absolute value is $|a + j * b| = \sqrt{a^2 + b^2}$



$$\left| \begin{array}{|c|c|c|c|c|c|c|c|} \hline -3 & -1 & 0 & 1 & 2 & 1 & 0 & -1 \\ \hline \multicolumn{1}{c}{\scriptstyle 0} & \multicolumn{1}{c}{\scriptstyle 1} & \multicolumn{1}{c}{\scriptstyle 2} & \multicolumn{1}{c}{\scriptstyle 3} & \multicolumn{1}{c}{\scriptstyle 4} & \multicolumn{1}{c}{\scriptstyle 5} & \multicolumn{1}{c}{\scriptstyle 6} & \multicolumn{1}{c}{\scriptstyle 7} \end{array} + j* \begin{array}{|c|c|c|c|c|c|c|c|} \hline 0 & -1 & 0 & 1 & 0 & -1 & 0 & 1 \\ \hline \multicolumn{1}{c}{\scriptstyle 0} & \multicolumn{1}{c}{\scriptstyle 1} & \multicolumn{1}{c}{\scriptstyle 2} & \multicolumn{1}{c}{\scriptstyle 3} & \multicolumn{1}{c}{\scriptstyle 4} & \multicolumn{1}{c}{\scriptstyle 5} & \multicolumn{1}{c}{\scriptstyle 6} & \multicolumn{1}{c}{\scriptstyle 7} \end{array} \right| = \begin{array}{|c|c|c|c|c|c|c|c|} \hline 3 & 1.4 & 0 & 1.4 & 2 & 1.4 & 0 & 1.4 \\ \hline \end{array}$$

Spectrogram

- All the N frequency values (frequency indices from 0 to N-1) are **real and positive** (abs!)



-3	-1	0	1	2	1	0	-1
0	1	2	3	4	5	6	7

+ j*

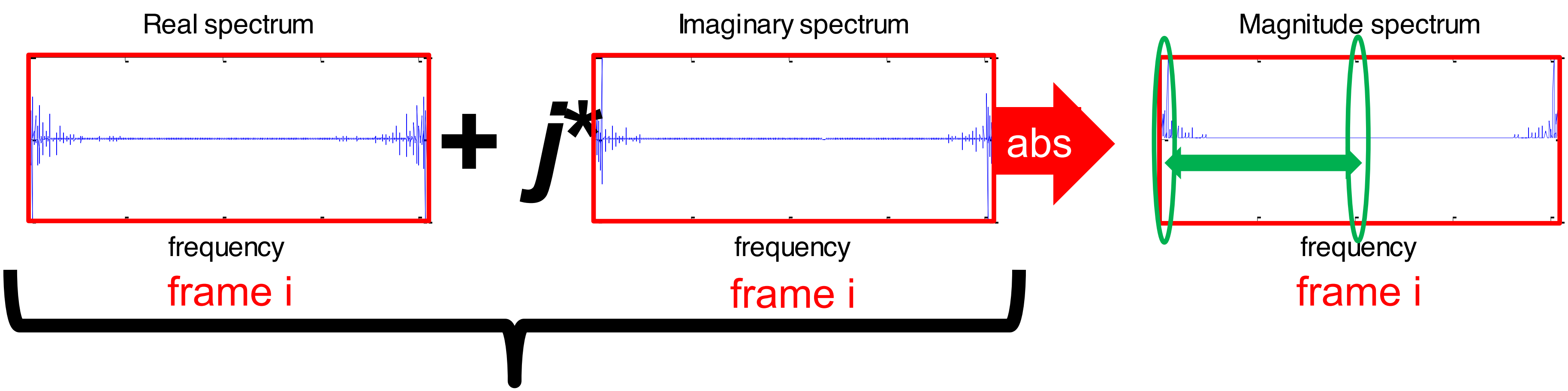
0	-1	0	1	0	-1	0	1
0	1	2	3	4	5	6	7

=

3	1.4	0	1.4	2	1.4	0	1.4
0	1	2	3	4	5	6	7

Spectrogram

- Frequency indices from 0 to floor(N/2) are the **unique frequency values** (with DC and pivot)



-3	-1	0	1	2	1	0	-1
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*

+ j*

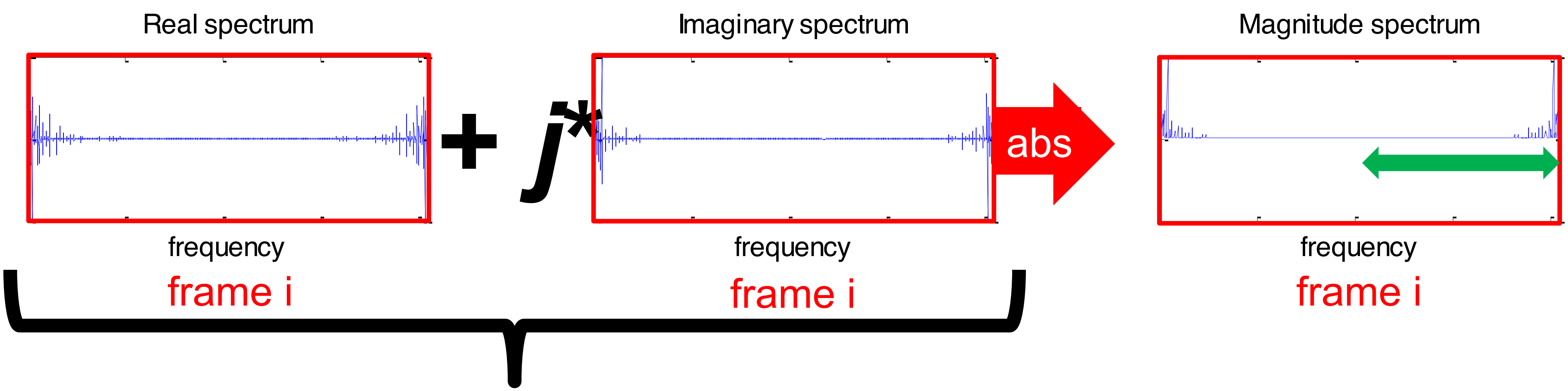
0	-1	0	1	0	-1	0	1
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*

=

3	1.4	0	1.4	2	1.4	0	1.4
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*

Spectrogram

- Frequency indices from floor(N/2)+1 to N-1 are the **mirrored frequency values**



-3	**-1**	**0**	**1**	**2**	**1**	**0**	**-1**
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*

+ *j**

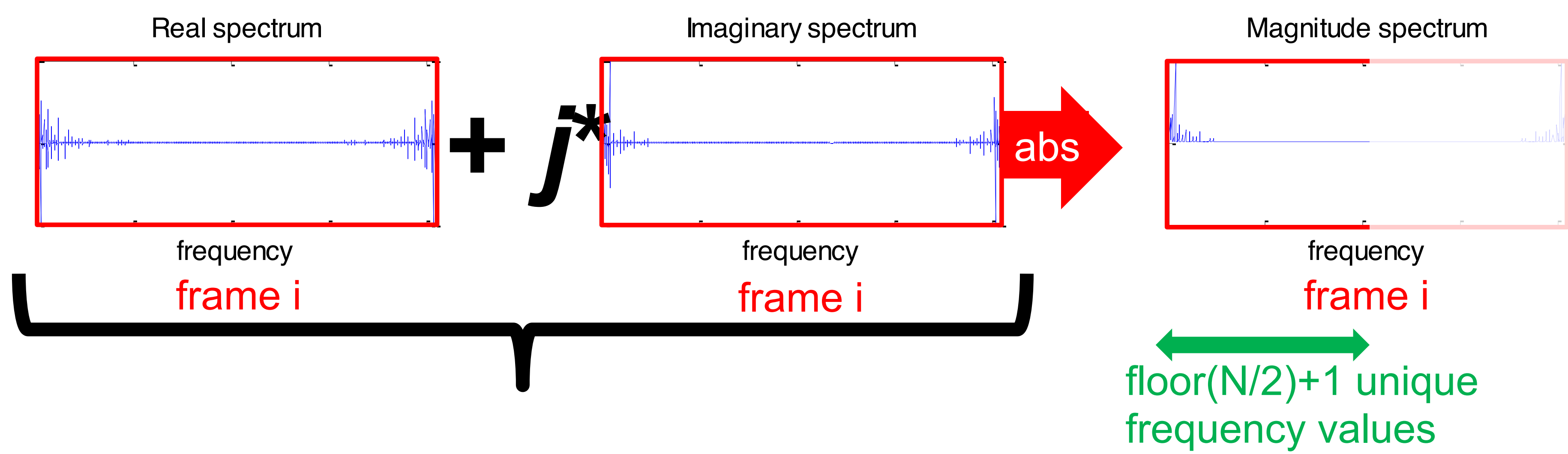
0	**-1**	**0**	**1**	**0**	**-1**	**0**	**1**
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*

=

3	**1.4**	**0**	**1.4**	**2**	**1.4**	**0**	**1.4**
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*

Spectrogram

- Since they are redundant, we can discard the frequency values from floor(N/2)+1 to N-1



-3	-1	0	1	2	1	0	-1
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*

+ j*

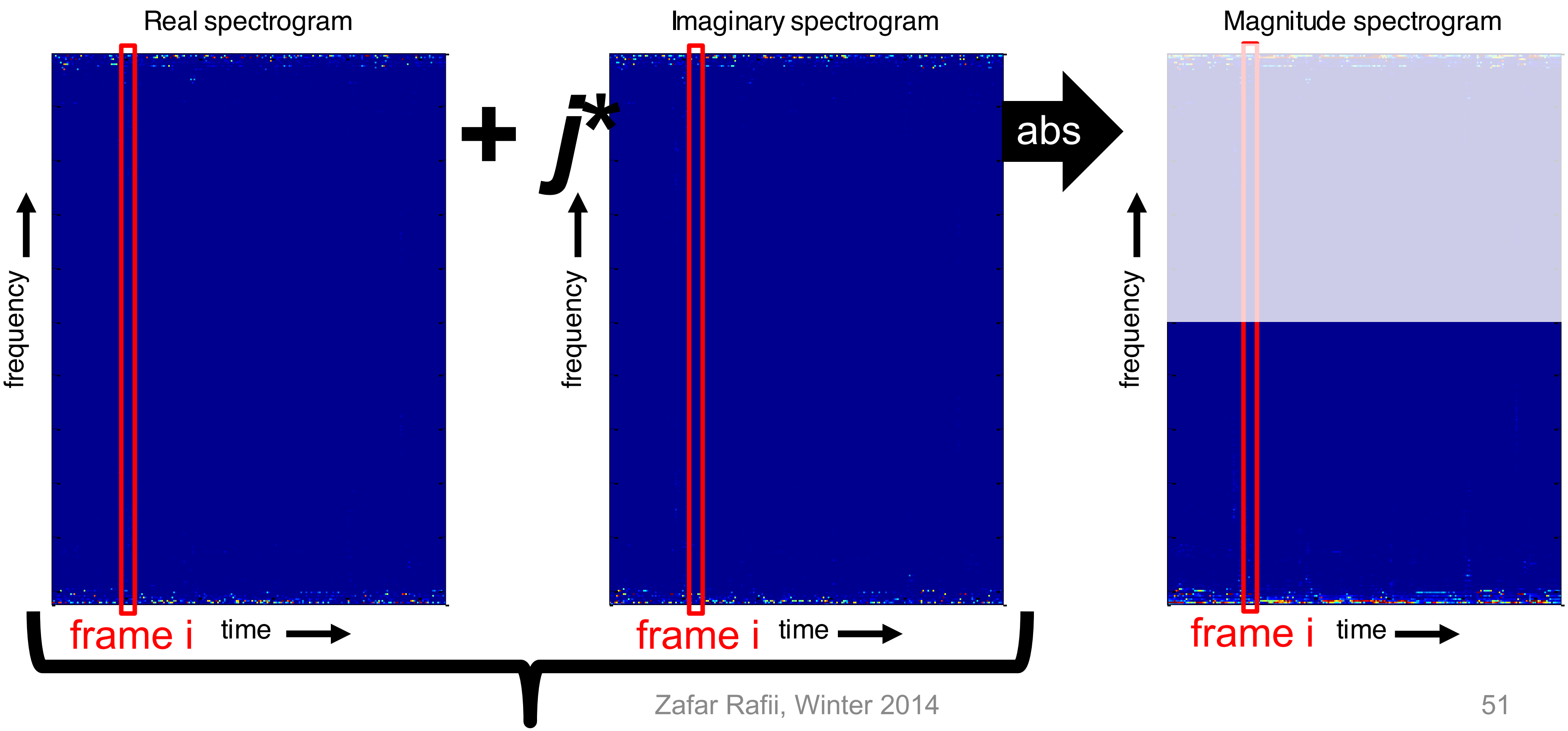
0	-1	0	1	0	-1	0	1
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*

=

3	1.4	0	1.4	2	1.4	0	1.4
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*

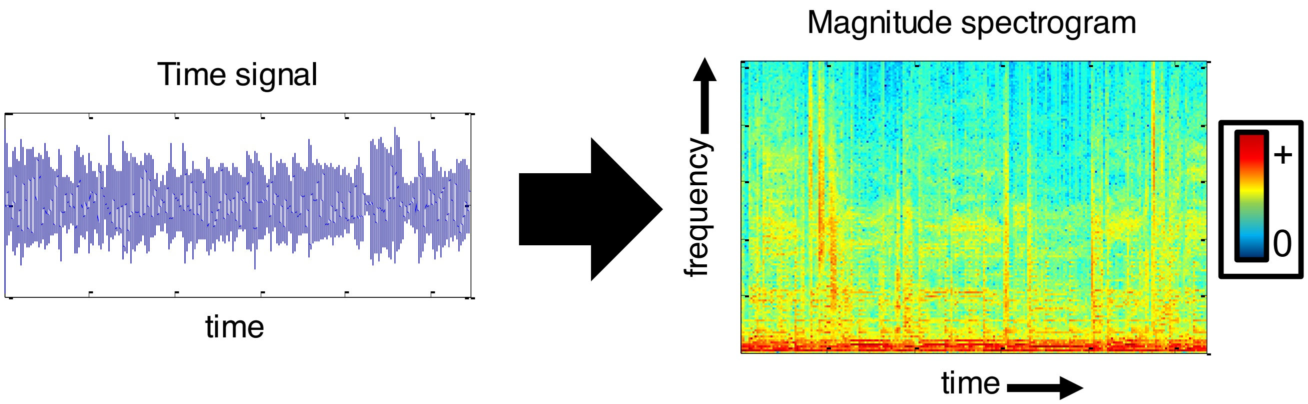
Spectrogram

- The spectrogram has therefore **floor(N/2)+1 unique frequency values** (with DC and pivot)



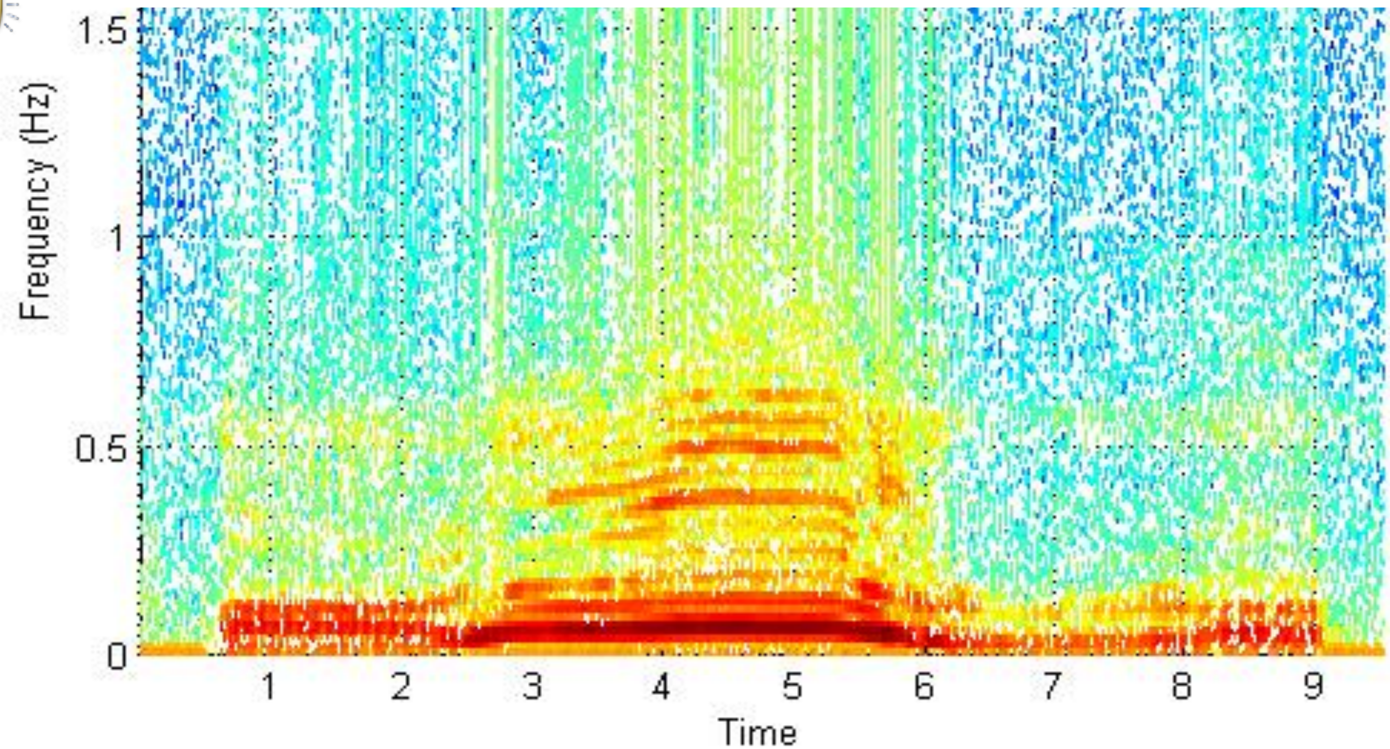
Spectrogram

- Why the magnitude spectrogram?
  – Easy to visualize (compare with the STFT)
  – Magnitude information more important
  – Human ear less sensitive to phase



The Spectrogram

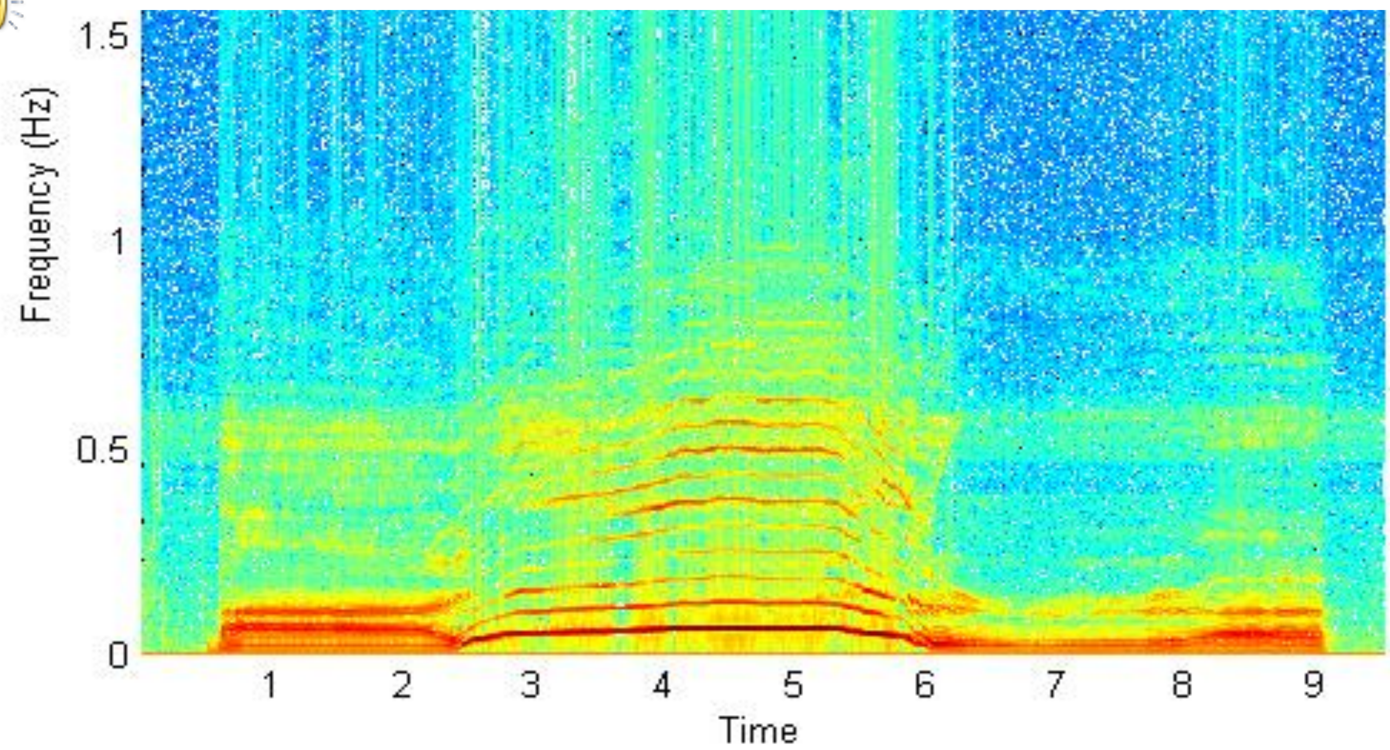
spectrogram(y,256,128,256,fs,'yaxis');



- A series of short term DFTs
- Typically just displays the magnitudes in X of the frequencies up to ½ sampe rate
- There is a **spectrogram** function in matlab

The Spectrogram

spectrogram(y,1024,512,1024,fs,'yaxis');



- A series of short term DFTs
- Typically just displays the magnitudes in X of the frequencies up to ½ sampe rate
- There is a **spectrogram** function in matlab