Computational Methods (PHYS 2030)

Instructors: Prof. Christopher Bergevin (cberge@yorku.ca)

Schedule: Lecture: MWF 11:30-12:30 (CLH M)

Website: http://www.yorku.ca/cberge/2030W2018.html

York University
Winter 2018
Lecture 24

Reminder

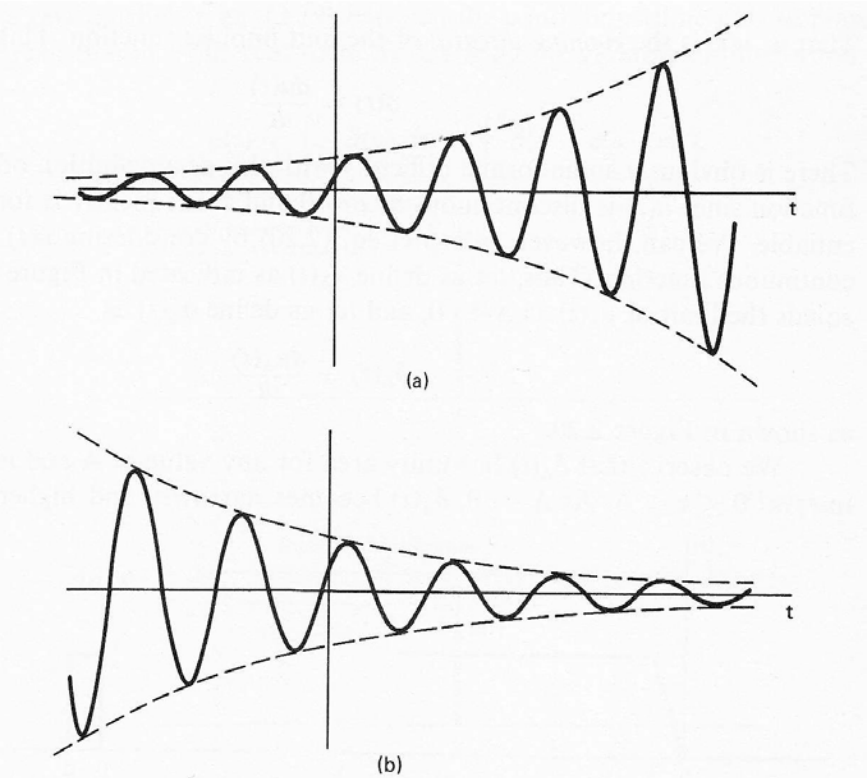


Figure 2.17 (a) Growing sinusoidal signal $x(t) = Ce^{rt}\cos(\omega_0 t + \theta)$, $r > 0$; (b) decaying sinusoid $x(t) = Ce^{rt}\cos(\omega_0 t + \theta)$, $r < 0$.

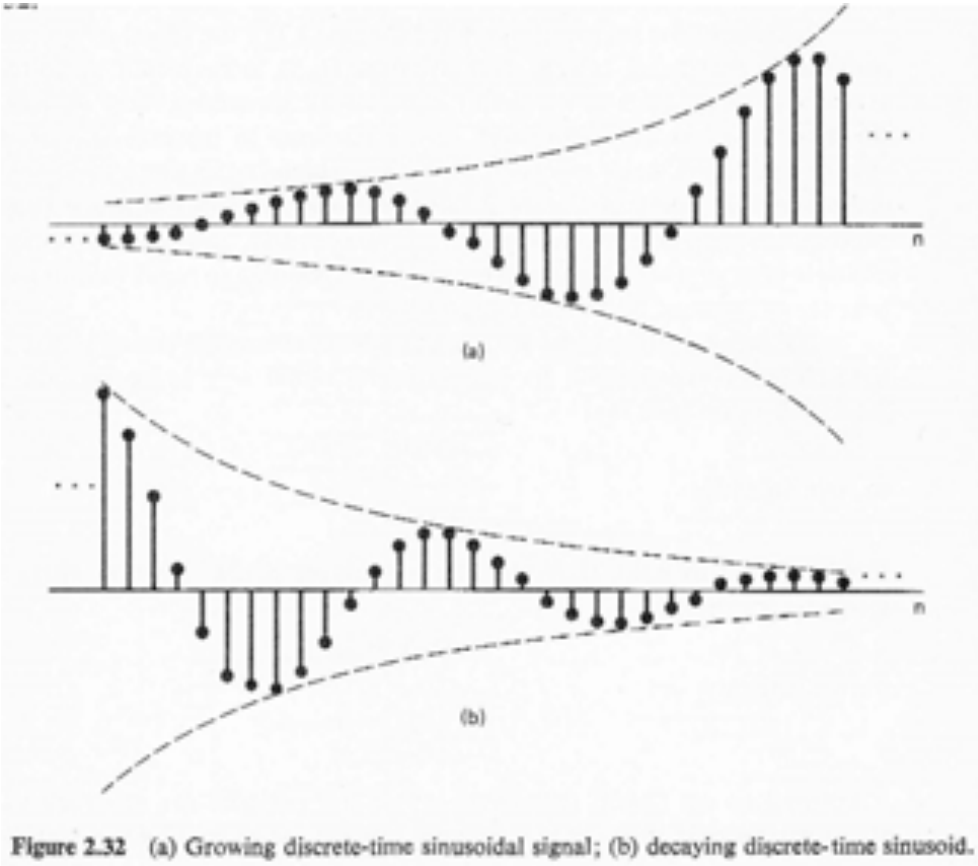


Figure 2.32 (a) Growing discrete-time sinusoidal signal; (b) decaying discrete-time sinusoid.

Continuous signal (analog)

Discretized signal (digital)

Question: Does Fourier analysis 'care' whether things are continuous or discrete?

Yes & no (we'll come back to this now)

Oppenheim & Willsky

Discrete Fourier transforms

- Many signals we deal with computationally are 'digital'. That is, the signal is discretely sampled at intervals Δt at a **S**ample **R**ate (SR, $1/\Delta t$) [Hz].

Our sampled signal: $f(m\Delta t), m = 0, 1, \ldots, N-1$

Fourier transform: $$g(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{-i\omega t}\, dt$$

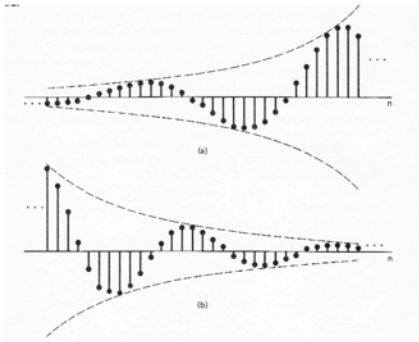


Figure 2.32 (a) Growing discrete-time sinusoidal signal; (b) decaying discrete-time sinusoid.

Since we have the measurements $f(m\Delta t)$, the integral can be performed numerically, by the trapezoid rule, for example. But there are some problems. First, we didn't take any data points before we started taking data. That is, *we don't have data before t=0!* And we don't have *continuous* data, but only data at the times $m\Delta t$! Oops. Maybe this isn't going to be so easy, after all.

- Think back to the notion of 'information': An infinite interval has infinite info. But since we sample a finite interval, we have limited info. Thus perhaps the best we can do is make an approximation given what we have in hand

Devries (1994)

Discrete Fourier transforms (DFT)

- Assume we have an interval long enough that all the 'interesting behavior' is contained within our sampled waveform. Then over the interval $0 < t < T$, the transform is:

$$f(t) = \sum_{n=-\infty}^{\infty} c_n e^{i2\pi nt/T} \qquad c_n = \frac{1}{T}\int_0^T f(t)e^{-i2\pi nt/T}\,dt$$

- This representation is periodic with period T. That is, we implicitly assume a periodic boundary condition [we'll come back to this soon in the context of 'quantizing frequency']

- Defining a new (more intuitive) variable and applying the trapezoid rule:

Discrete Fourier transform

$$\Delta\omega = \frac{2\pi}{T} \qquad g(n\Delta\omega) = \sum_{m=0}^{N-1} f(m\Delta t)e^{-in\Delta\omega m\Delta t} = \sum_{m=0}^{N-1} f(m\Delta t)e^{-i2\pi mn/N}$$

→ Computationally, this is what we crunch numerically and use for digital signal processing

Devries (1994)

Reminder: FFT

- A means to efficiently compute a DFT

An Algorithm for the Machine Calculation of Complex Fourier Series
Author(s): James W. Cooley and John W. Tukey
Source: *Mathematics of Computation*, Vol. 19, No. 90 (Apr., 1965), pp. 297-301

1. It has a low operation count: $O(N \log N)$.
2. It finds the transform on an interval $x \in [-L, L]$. Since the integration kernel $\exp(ikx)$ is oscillatory, it implies that the solutions on this finite interval have periodic boundary conditions.
3. The key to lowering the operation count to $O(N \log N)$ is in discretizing the range $x \in [-L, L]$ into 2^n points, i.e. the number of points should be $2, 4, 8, 16, 32, 64, 128, 256, \cdots$.
4. The FFT has excellent accuracy properties, typically well beyond that of standard discretization schemes.

→ EXspecREP3.m lets us examine computationally the spectra (via the FFT) of some common/intuitive waveforms

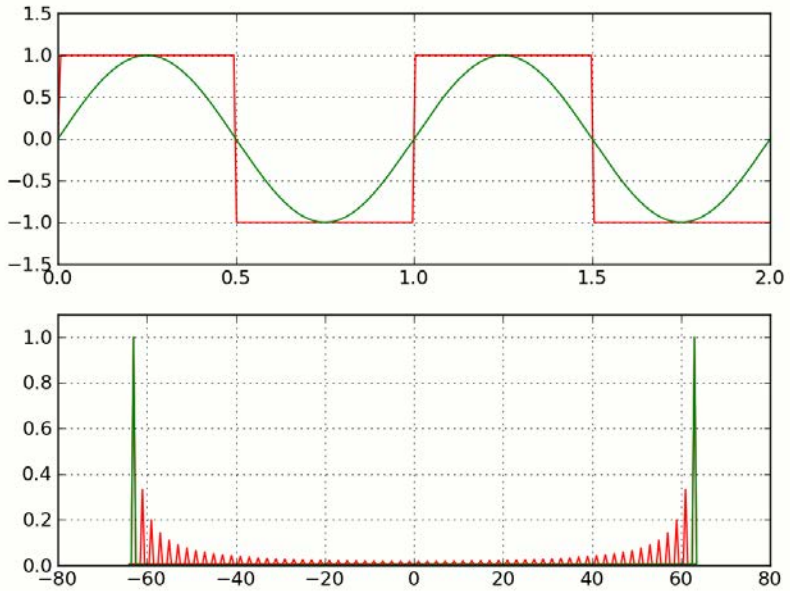
Kutz (2012)

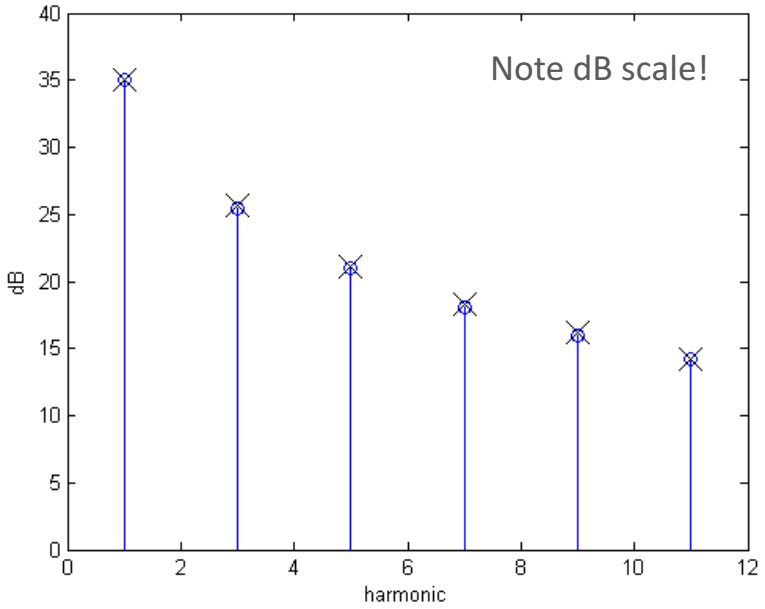
`fft.m` vs. `rfft.m`

- For situations where the function is purely real (e.g., a sampled waveform), the Fourier transform shows a key symmetry – *positive and negative frequencies are mirror images*

If $f(t)$ is real, then $\Re g(\omega)$ is even and $\Im g(\omega)$ is odd;

→ Thereby, we can toss out half the DFT without any loss of information





- `rfft.m` does this, as well as normalizes the amplitudes $\mathcal{F}[f(\alpha t)] = \frac{1}{|\alpha|} g(\frac{\omega}{\alpha})$

wikipedia (square wave)
http://www.johnloomis.org/ece303L/lab4/series_demo/Fourier_Square.html

DFT Limitations: Shannon-Nyquist Sampling Theorem

- "In the field of digital signal processing, the *sampling theorem* is a fundamental bridge between continuous signals (*analog* domain) and discrete signals (*digital* domain)." [wikipedia (Nyquist–Shannon sampling theorem)]

- Basically, there are two crucial parameters with respect to sampling (i.e., how much information are you capturing):

  - Sample rate (SR) – Limits how 'fast' you can pick off frequencies. If you are too slow, you won't be able to pick off higher frequencies.

  - Window length (N) – Put another way, the number of samples. This will ultimately contribute to determining the 'bin' frequencies of the DFT.

    → In a nutshell, the highest frequency you can 'capture' in your DFT is SR/2. Anything faster than this will cause *aliasing*

EXspecREP3.m

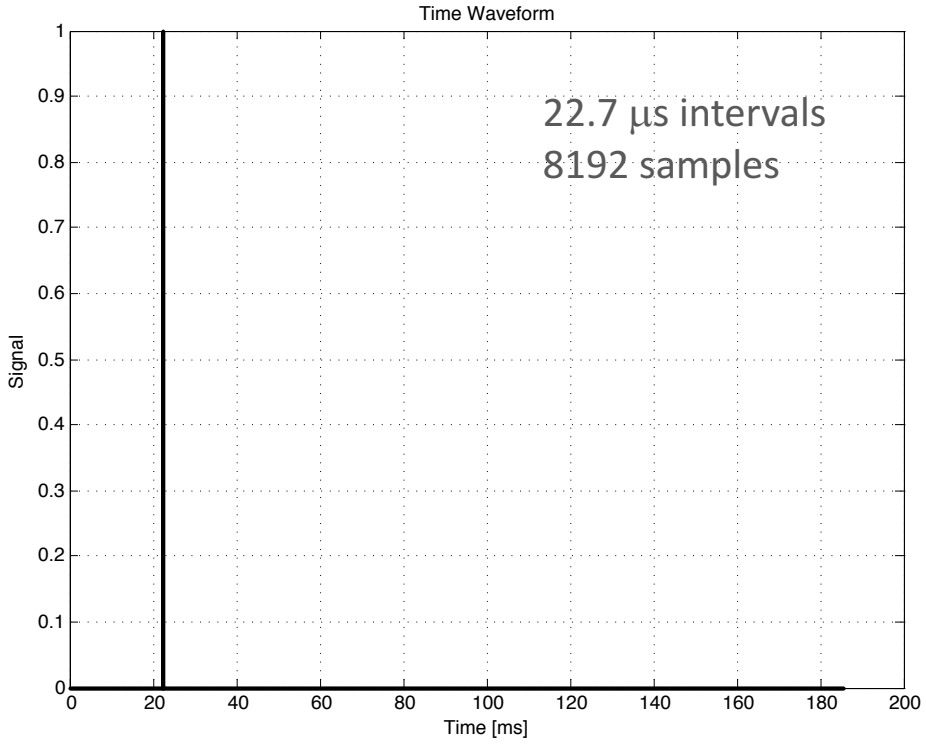
$$\Delta\omega = \frac{2\pi}{T}$$

```
SR= 44100;          % sample rate [Hz]
Npoints= 8192;      % length of fft window (# of points)
freq= [0:Npoints/2];    % create a freq. array (for FFT bin labeling)
freq= SR*freq./Npoints;
```

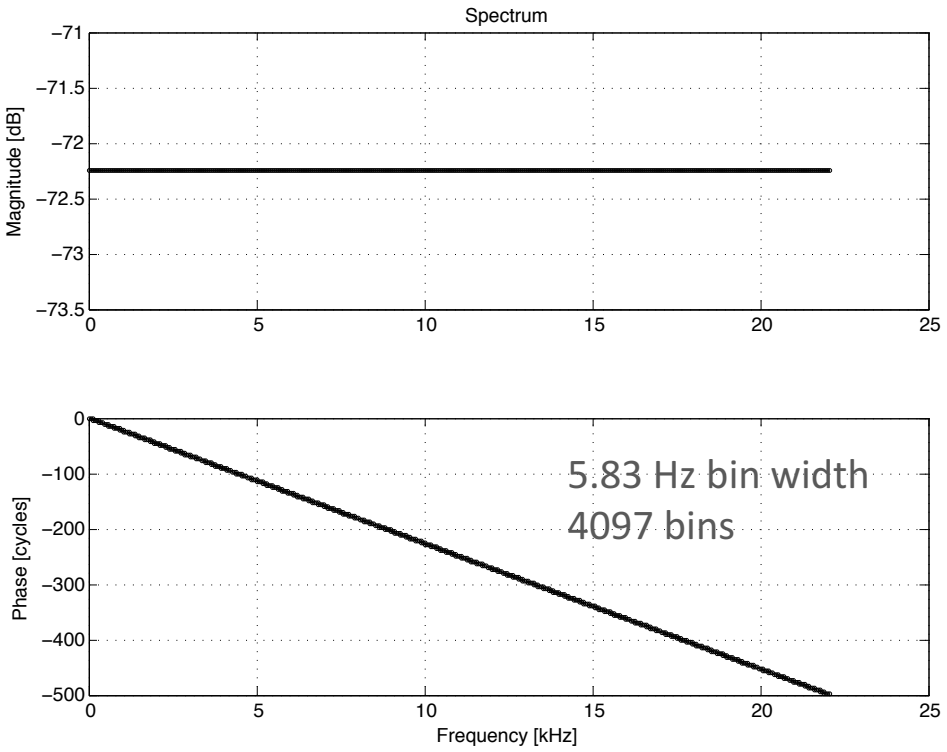
DFT Limitations: Shannon-Nyquist Sampling Theorem

```
SR= 44100;          % sample rate [Hz]
Npoints= 8192;      % length of fft window (# of points)
freq= [0:Npoints/2];    % create a freq. array (for FFT bin labeling)
freq= SR*freq./Npoints;
```

Time domain



Spectral domain

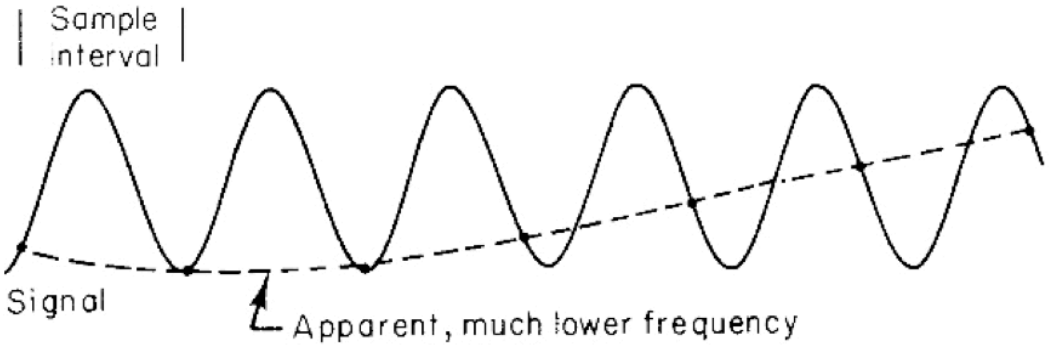


8192/44100 = 0.186 s
1/44100= 22.7 μs

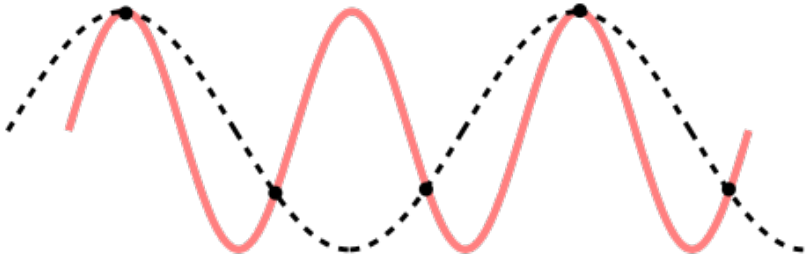
44100/2 = 22.1 kHz

Aside: Aliasing

→ In a nutshell, the highest frequency you can 'capture' in your DFT is SR/2. Anything faster than this will cause *aliasing*



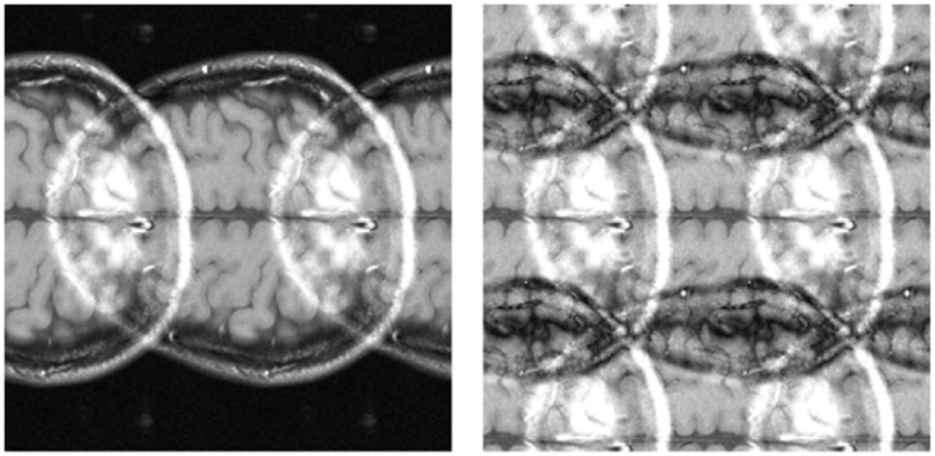
- '*Undersampling*' (i.e., too slow) misrepresents the waveform!



wikipedia (Nyquist–Shannon sampling theorem)

- *Two is the magic number*: You need to sample at least twice as fast.
  [the factor of two is associated with the symmetry between positive and negative frequencies of the FFT]

- Otherwise *artifacts* can occur
  (this happens in a wide variety of contexts where DFTs are used, such as MRI)



Hobbie & Roth (2007)

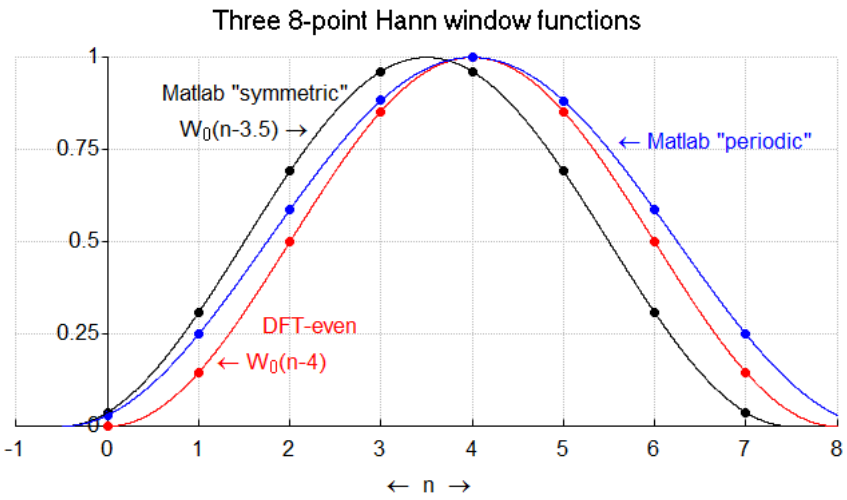
Windowing

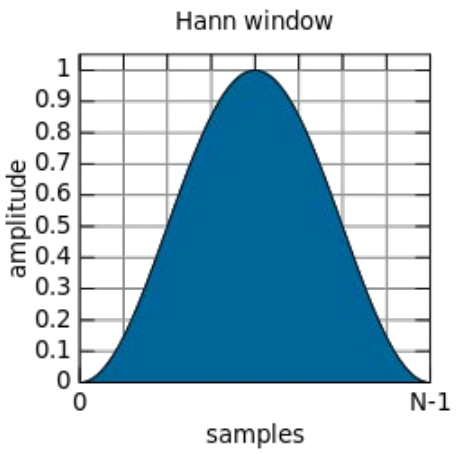
- Sometimes you have control over the signals of interest (e.g., you create them). Other times you do not (e.g., you are measuring something totally from an external source)

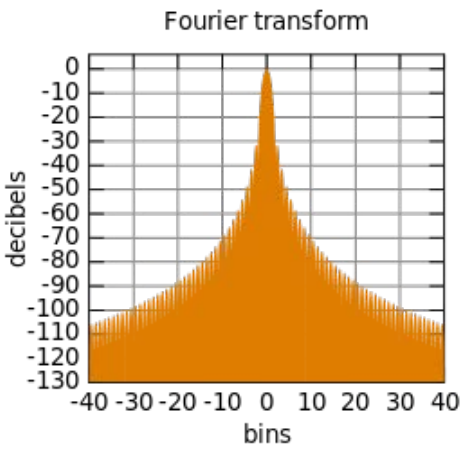
- While you can control your sample rate and window length (i.e., how many samples you collect), this subtle distinction can have big effects upon a spectral representation

- One way to work around this is by means of *windowing*

Basic idea: Multiply your time waveform by a (well-chosen) function that smoothly 'tamps down' the measurements at the start and end of the interval. You might lose a bit of information, but the trade-off can be worth it spectrally





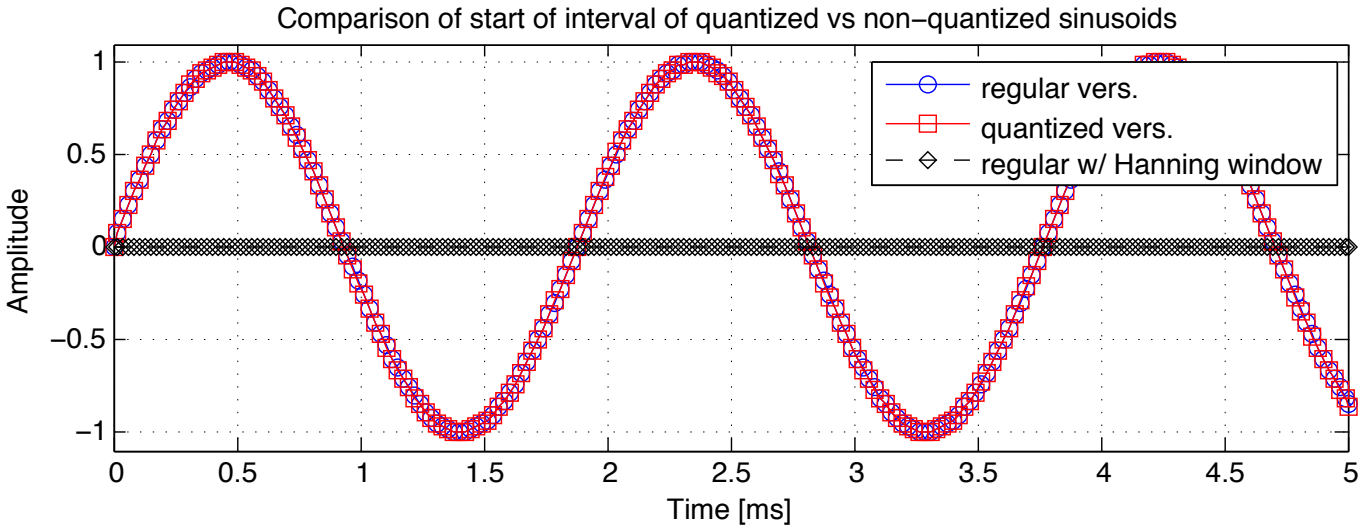


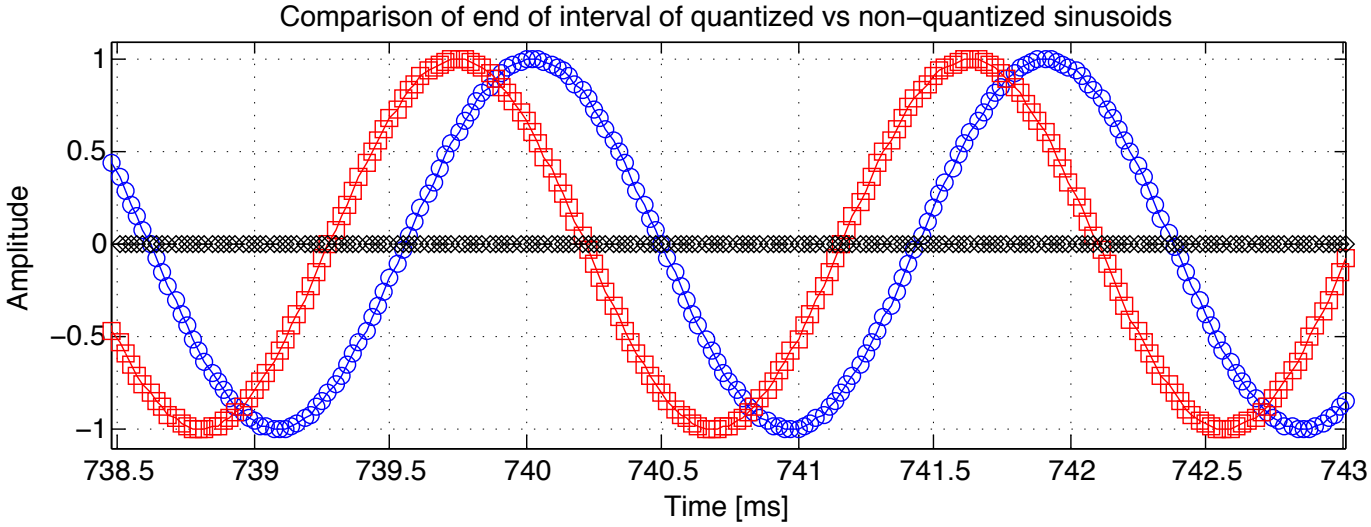
wikipedia (window function)

```
% ### EXquantizeF.m ###       11.04.14
% code to demonstrate effects/necessity of quantizing freq. for discrete FFT
% [that is, making sure the signal is periodic re the interval]
clear;
% --------------------------------
f= 531.4;          % freq. {1000}
SR= 44100;          % sample rate {44100}
Npoints= 32768;     % length of fft window (# of points) [should ideally be 2^N] {8192}
% --------------------------------
% +++
dt= 1/SR;  % spacing of time steps
freq= [0:Npoints/2];    % create a freq. array (for FFT bin labeling)
freq= SR*freq./Npoints;
% +++
% quantize the freq. (so to have an integral # of cycles)
df = SR/Npoints;
fQ= ceil(f/df)*df;   % quantized natural freq.
disp(sprintf('specified freq. = %g Hz', f));
disp(sprintf('quantized freq. = %g Hz', fQ));
% +++
% create an array of time points, Npoints long
t=[0:1/SR:(Npoints-1)/SR];
% +++
w= sin(2*pi*f*t);   % non-quantized version
wQ= sin(2*pi*fQ*t); % quantized version
wH= hanning(Npoints).*w';   % also window the non-quantized version...
% +++
% plot time waveforms for comparison
figure(1); clf;
subplot(211)
plot(t*1000,w,'o-'); hold on; grid on;
plot(t*1000,wQ,'rs-');
plot(t*1000,wH,'k--d');
axis([0 5 -1.1 1.1]); legend('regular vers.','quantized vers.','regular w/ Hanning window')
xlabel('Time [ms]'); ylabel('Amplitude')
title('Comparison of start of interval of quantized vs non-quantized sinusoids')
subplot(212)
plot(t*1000,w,'o-'); hold on; grid on;
plot(t*1000,wQ,'rs-')
plot(t*1000,wH,'k--d')
axis([t(end-200)*1000 t(end)*1000 -1.1 1.1])
xlabel('Time [ms]'); ylabel('Amplitude')
title('Comparison of end of interval of quantized vs non-quantized sinusoids')
% +++
% now plot spectra for comparison
figure(2); clf;
plot(freq,db(rfft(w)),'o-','MarkerSize',3); hold on;
plot(freq,db(rfft(wQ)),'rs-','MarkerSize',4)
plot(freq,db(rfft(wH)),'k--d','MarkerSize',5)
xlabel('freq. [Hz]'); ylabel('magnitude [dB]');
grid on; axis([0 1.5*f -350 10])
legend('non-quantized version','quantized version','Windowed non-quantized','Location','SouthWest')
```

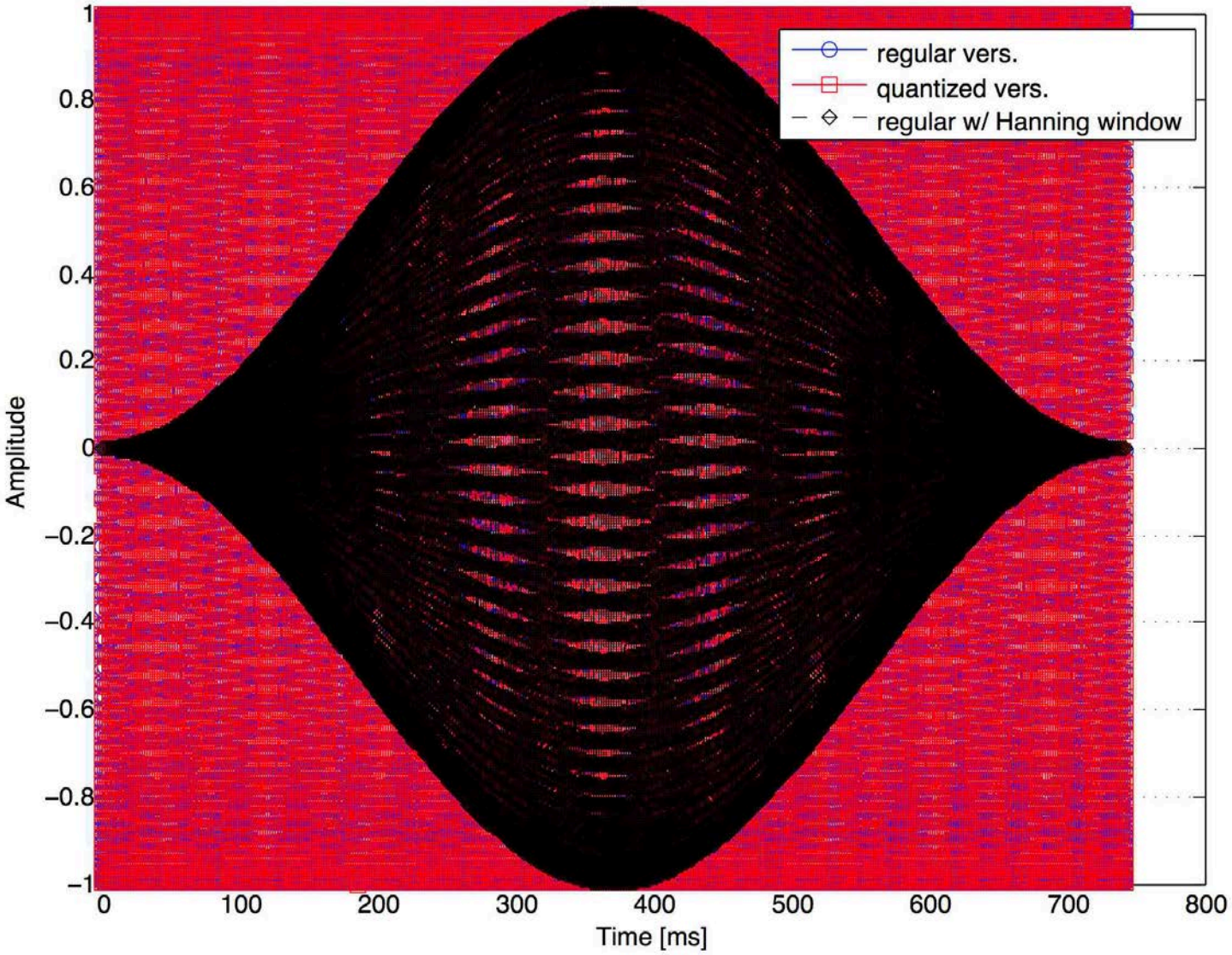
- Demonstrates two key concepts:
  - a discrete FFT is highly sensitive to the assumption about a periodic boundary condition
  - when you don't have direct control over the signals you measure, 'windowing' can help reduce this sensitivity

```
f= 531.4;          % freq. [Hz]
SR= 44100;          % sample rate [Hz]
Npoints= 32768;     % length of fft window (# of points)
```



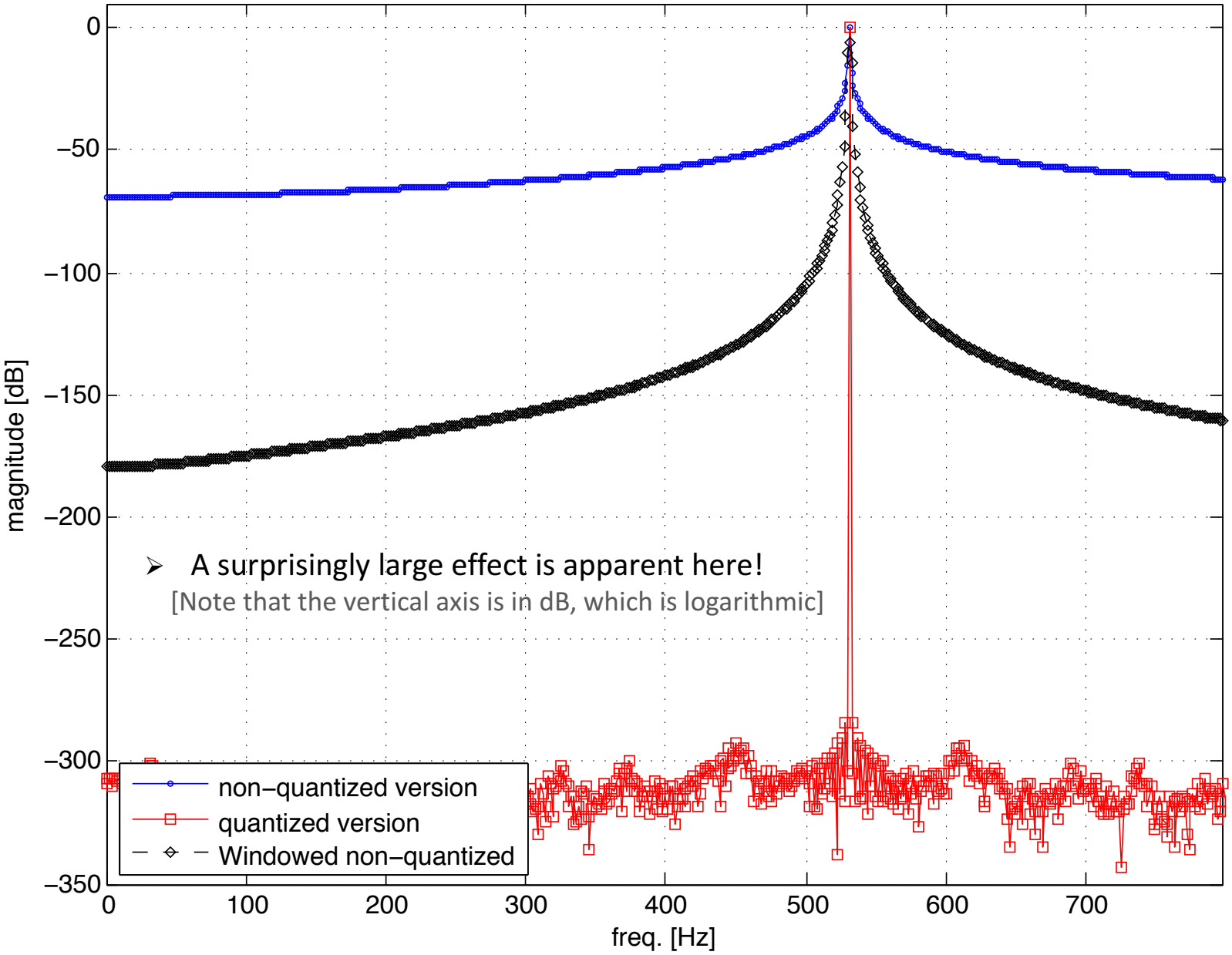


Difference is quite subtle:
→ Is there an integer number of periods in the interval?



- Windowing simply smoothly ‘tamps down’ the signal at both ends

- Useful when you don’t have control over the signal in some fashion



- A surprisingly large effect is apparent here!

[Note that the vertical axis is in dB, which is logarithmic]

DC vs AC

- DC (or 'direct current') typically refers to static conditions (i.e., $|\omega|=0$)
- AC (or 'alternating current') typically refers to sinusoidal conditions (i.e., $|\omega|>0$)
- This simple distinction has a wide variety if implication throughout physics and engineering

Ex. Ohm's law

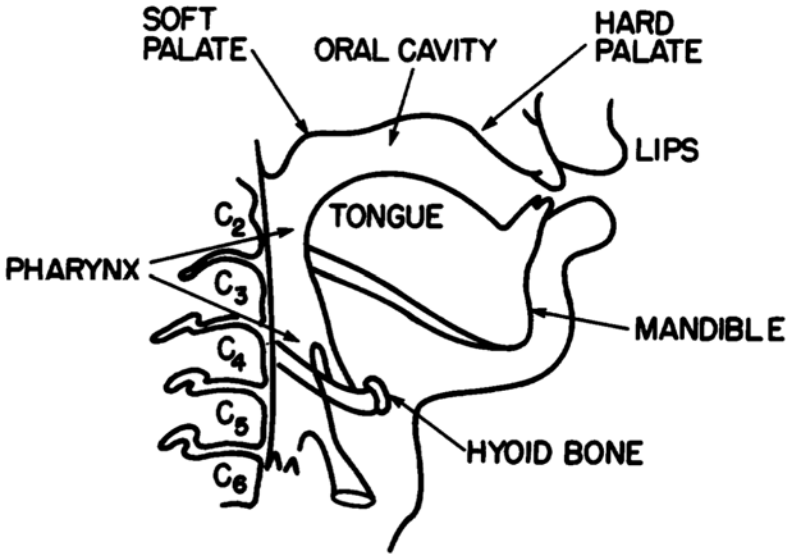
$$V = IR$$ DC version

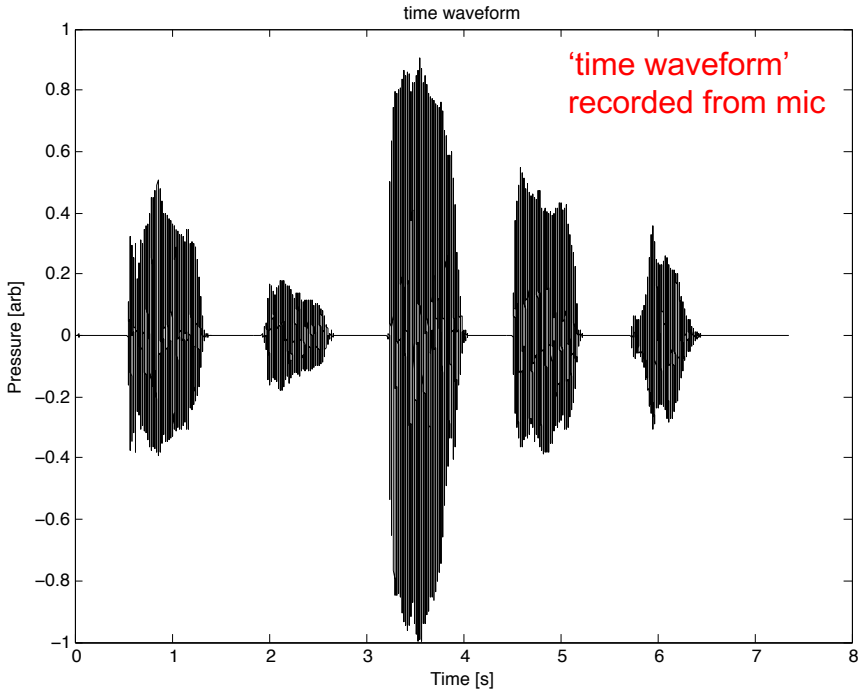
$$V = IZ$$ AC version
(all variables are complex Fourier coefficients)

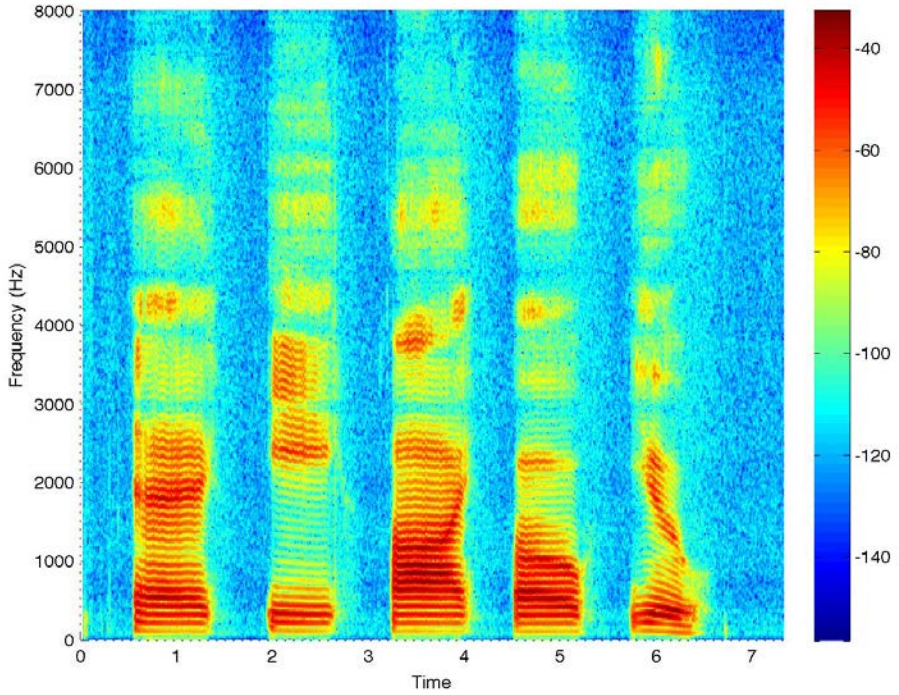
→ Here, Z is known as the *impedance* (we'll come back to this)

Spectrograms

EXspectrogram.m







Spectrogram =
Short Time Fourier Transform (STFT)

Stevens (2000)

STFT

- Basic idea is to compute DFT over short intervals (i.e., short segments from a longer waveform) and thereby see how frequency content is changing with time

- Clearly there is a tradeoff between time and frequency resolution

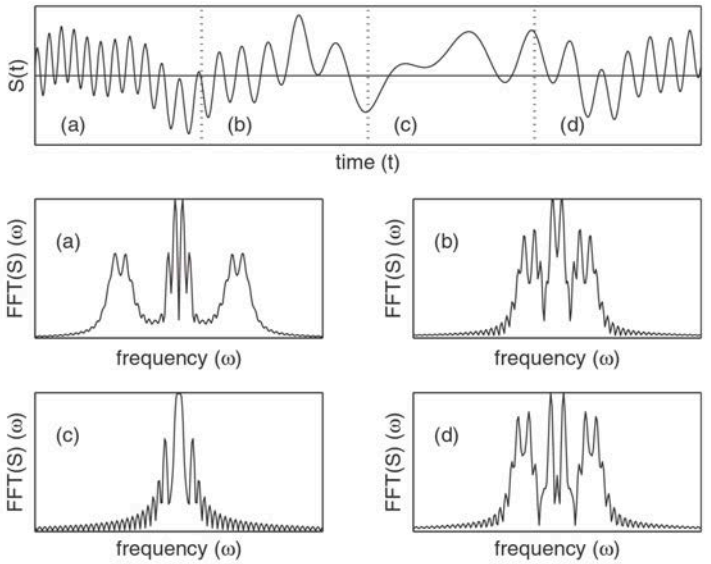
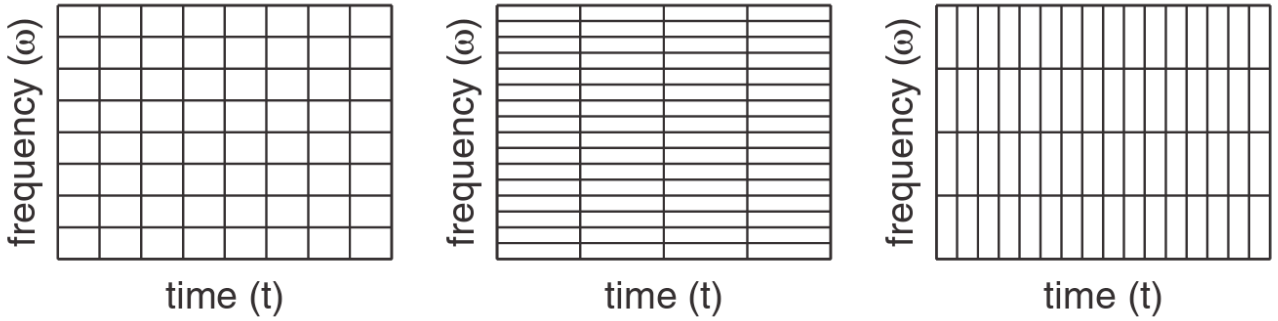


Figure 13.14: Signal $S(t)$ decomposed into four equal and separate time-frames (a), (b), (c) and (d). The corresponding normalized Fourier transform of each time-frame $\hat{S}(\omega)$ is illustrated below the signal. Note that this decomposition gives information about the frequencies present in each smaller time-frame.



EXspectrogram.m

```
P.windowL= 2048;    % length of window segment for FFT {2048}
P.overlap= 0.8;     % fractional overlap between window, from 0 to 1 {0.8}
pts= round(P.windowL*P.overlap);  % convert fractional overlap to # of points
spectrogram(A,blackman(P.windowL),pts,P.windowL,P.SR,'yaxis');
```

'overlapping' and windowing can help

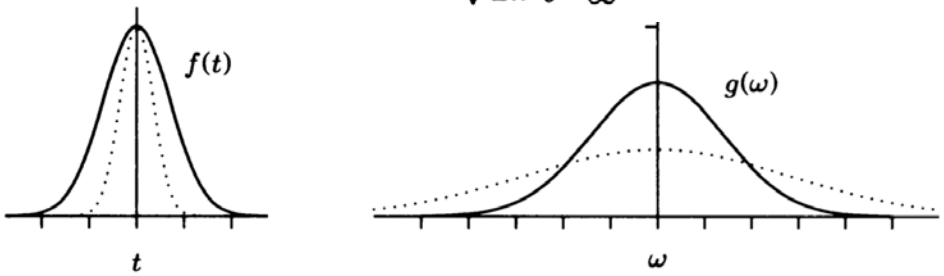
Kutz (2012)

Summary

Information can represented in different ways by different ‘basis functions’

$$f(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} g(\omega) e^{i\omega t} d\omega$$

$$g(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt$$



Uncertainty – Localization in one domain means broadening in the other

Fourier series

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos nt + \sum_{n=1}^{\infty} b_n \sin nt$$

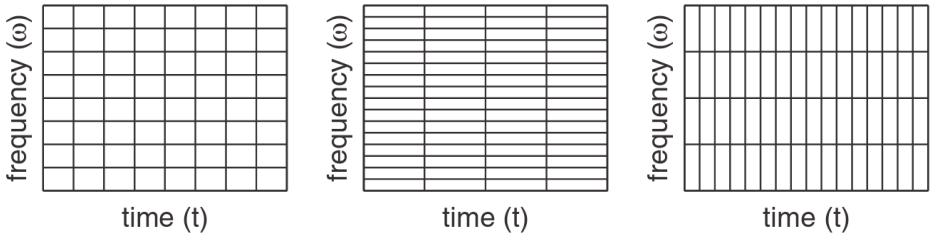
Fourier transform

$$\mathcal{F}[f(t)] = g(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt$$

Discrete Fourier transform

$$g(n\Delta\omega) = \sum_{m=0}^{N-1} f(m\Delta t) e^{-in\Delta\omega m\Delta t} = \sum_{m=0}^{N-1} f(m\Delta t) e^{-i2\pi mn/N}$$

Three sides of the same coin



Practically, there will be various trade-offs

Exercises

- Fiddle around with `EXquantizeF.m`. How are things affected if you change various parameters? What if you make the window longer or shorter? Or try different windows (e.g., `blackman,hamming, boxcar, etc...`).

- What does it mean for the phase to be 'unwrapped'?

- Using `EXspecREP3.m`, determine the difference between and exponentially decaying sin versus a cos. How does such relate back to differences in the initial conditions of a harmonic oscillator?

- Using `EXspecREP3.m`, for one or more sinusoids, specify a frequency higher than twice the sample rate and observe the variety of aliasing effects that arise

- Solve the van der Pol oscillator for some period of time. Extract a segment of the steady-state response (i.e., after settling into the limit cycle) and compute the associated DFT. What does is look like? What does this tell you about the oscillator?

- For what cases is the Fourier transform not symmetric about zero frequency?

- Using `EXspecREP3.m`, distinguish how AC vs DC components to a signal affect the spectrum.