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
Connection to Fourier Transforms

Consider the Fourier transform of the (1-D) convolution:

\[ \mathcal{F}[p \otimes q] = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} [p \otimes q] e^{-i\omega t} \, dt \]

\[ = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \left[ \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} p(\tau) q(t-\tau) \, d\tau \right] e^{-i\omega t} \, dt \]

\[ = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} p(\tau) \left[ \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} q(t-\tau) e^{-i\omega t} \, dt \right] d\tau. \]

Making use of the ‘shifting property’, the term in the square brackets is:

\[ \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} q(t-\tau) e^{-i\omega t} \, dt = e^{-i\omega \tau} Q(\omega) \]

\( Q(\omega) \) is the Fourier transform of \( q(t) \)

\[ \mathcal{F}[p \otimes q] = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} p(\tau) e^{-i\omega \tau} Q(\omega) \, d\tau \]

\[ = P(\omega)Q(\omega), \]

\( P(\omega) \) is the Fourier transform of \( p(t) \)

\[ \mathcal{F}[p \otimes q] = \mathcal{F}[p] \mathcal{F}[q] \]

Devries (1994)
Simple but powerful idea: Convolution in the time domain is simply a multiplication in the spectral domain.

\[ \mathcal{F}[p \otimes q] = \mathcal{F}[p] \mathcal{F}[q] \]

Door swings both ways: From the output, if we know the impulse response, we can deconvolve (i.e., divide in spectral domain) to get the original input!

\[ V_{out} = V_{in} \otimes r, \quad \mathcal{F}[V_{out}] = \mathcal{F}[V_{in} \otimes r] = \mathcal{F}[V_{in}] \mathcal{F}[r] \]

\[ \mathcal{F}[V_{in}] = \frac{\mathcal{F}[V_{out}]}{\mathcal{F}[r]} \]

\[ V_{in}(t) = \mathcal{F}^{-1} \left[ \frac{\mathcal{F}[V_{out}]}{\mathcal{F}[r]} \right] \]
If you have some estimate of your PSF, you can ‘correct’ your image very efficiently in the frequency domain.
Biomechanically, middle ear acts as an ‘impedance matcher’
Ex. Middle ear  

- Eardrum is a thin, tent-like membrane

- Sound-induced motion is surprisingly complex

- Finite element models are one approach... (but require lots of assumptions about parameters)
Ex. Middle ear

- but a surprisingly effective approach is much simpler: ‘lumped elements’
  (i.e., electrical circuit analog)

Various circuit elements are acoustic analogs (e.g., capacitance represents springiness of air compression of closed cavity)
Ex. Middle ear

To decent first order, model captures many experimental features of sound transmission.

- Essentially amounts to the system’s impulse response (or the closely related ‘transfer function’)

We’ve characterized the ‘filtering’ properties of the middle ear!

$$V_{out}(t) = \int_{-\infty}^{\infty} V_{in}(\tau)r(t - \tau) \, d\tau = V_{in} \otimes r.$$
Exercises

- What happens when the position of the resistor and capacitor are swapped?

\[
\frac{dV_{out}}{dt} + \frac{V_{out}}{RC} = \frac{V_{in}}{RC}
\]

- Confirm that the given solution does indeed satisfy the ODE.

\[
V_{out}(t) = e^{-t/RC} \left[ \int_{-\infty}^{t} e^{\tau/RC} V_{in}(\tau) d\tau + C_1 \right]
\]

- If you wanted to build a low-pass filter with a particular ‘cutoff frequency’, what would you pick for R and C? How exactly might you construct the circuit (e.g., what gets soldered to what)?

- For image processing, what sort of kernel would you use for ‘edge detection’? How do you think this relates to a ‘mask’ in the spectral domain?

- Using EXconvolve1.m, try varying the waveforms and observe the different patterns that arise. Are they consistent with what you expected?
In some cases, you have control over the ‘signal’ to be measured. That is, you control something going ‘out’ and you are looking for the response coming back in. Such commonly arises when measuring ‘transfer functions’ in network analysis.

Ex. Impulse voltage across an electric circuit

Ex. Speaker outputting a signal which is then measured back in a microphone
Ex. Quantizing frequency

In these cases, if using sinusoidal stimuli, it pays to be smart about exactly what stimulus parameters you use.

- **SR** – Sample Rate
  (e.g. 44.1 kHz)
- **N** – # of time points for FFT window
  (e.g. 8192)
- **f** – desired frequency [Hz]

\[ df \equiv \frac{SR}{N} \]

\[ f_Q = \left\lceil \left( \frac{f}{df} \right) \right\rceil \cdot df \]

'quantized' frequency to present

i.e., round up to nearest integer (the ceiling)

→ Slightly changing your output frequency can have a big effect!
% Code to show effects/necessity of quantizing freq.
% [NOTE - requires: db.m, rfft.m, and hanning.m]

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 = SR*(0:Npoints/2)./Npoints; % create a freq. array (for FFT bin labeling)
df = SR/Npoints; % quantize the freq. (so to have an integral # of cycles)
fQ = ceil(f/df)*df; % quantized natural freq.
disp(sprintf('specified freq. = %g Hz', f));
disp(sprintf('quantized freq. = %g Hz', fQ));

t=[0:1/SR:(Npoints-1)/SR]; % create an array of time points, Npoints long
w = sin(2*pi*f*t); % non-quantized version
wQ = sin(2*pi*fQ*t); % quantized version
wH = hanning(Npoints).*w; % could also apply a window too to 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('Note: non-quantized has arbitrary # of cycles in total FFT window, quantized has an integer #')
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')

% +++
% now plot fft of both for comparison
figure(2); clf;
plot(freq,db(rfft(w)),'o-','MarkerSize',3); hold on; grid on;
plot(freq,db(rfft(wQ)),'rs-','MarkerSize',4);
plot(freq,db(rfft(wH)),'k--d','MarkerSize',5);
xlabel('freq. [Hz]'); ylabel('magnitude [dB]'); axis([0 1.5*f -350 10])
legend('regular version','quantized version','regular version w/ Hanning window','Location','NorthWest')
Note: non-quantized has arbitrary # of cycles in total FFT window, quantized has an integer #
Such a small effect in the time domain has such a big effect in the spectral domain!

Artifactual noise floor could ‘mask’ many signals of interest!

Ultimately what we are doing here is ‘enforcing’ the assumption about the *periodic boundary condition* for the DFT.

**Lesson:** When you have control, use it (wisely)
Noise

- In many cases, there will be unwanted \textit{noise} (i.e., random fluctuations outside of our control) in the signals you are trying to measure

- Such arises from a variety of sources, and many classes of statistics attempt to deal with such head on.

- Ideally, noise is best dealt with at the source \textbf{when possible} (e.g., vibration isolation table for atomic force microscopy, 60 Hz ‘line noise’).

We’ve already seen many practical problems that deal with noise (e.g., regression analysis)
A useful empirical measure is the ‘signal to noise ratio’ (SNR), which quantifies the relative balance between ‘signal’ (i.e., useful information) and the noise.

Typically done via the spectral domain:

- In terms of power:
  \[ \text{SNR} = \frac{P_{\text{signal}}}{P_{\text{noise}}} \]

- In terms of amplitude:
  \[ \text{SNR} = \frac{P_{\text{signal}}}{P_{\text{noise}}} = \left( \frac{A_{\text{signal}}}{A_{\text{noise}}} \right)^2 \]

- In terms of dB:
  \[ \text{SNR}_{\text{dB}} = 10 \log_{10} \left[ \left( \frac{A_{\text{signal}}}{A_{\text{noise}}} \right)^2 \right] = 20 \log_{10} \left( \frac{A_{\text{signal}}}{A_{\text{noise}}} \right) \]
Averaging

- One way to deal with this is by means of averaging – By making repeated measurements, such can be combined to reduce noise and thus improve one’s SNR.

- Some forms of averaging are closely related to convolutions and the associated concept of correlations.

\[
p \odot q = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} p^*(\tau)q(t + \tau) \, d\tau
\]

Devries (1994)
Wikipedia (cross-correlation)
In many practical contexts, averaging comes in two different flavors: **temporal** and **spectral**

- **Temporal averaging** – Simply averaging repeated measures in the time domain. Essentially cross-correlating a signal with (a noisy/randomized version of) itself, hence a close connection to the autocorrelation.

- **Spectral averaging** – For every signal, compute (usually) the Fourier transform and subsequently average the magnitudes. Useful when there is no ‘phase locking’ to the incoming signal.

**Note:** An important point is that some degree of ‘information’ is inherently lost when averaging (i.e., you are tossing out some aspect of the data!)

![Image of signal averaging](image.png)
% clear
% --------------------------------
SR = 44100; % sample rate [44100]
Npoints = 8192; % length of fft window (# of points) [should ideally be 2^N] [8192]
f = 3.2; % 'stimulus' freq. [kHz]
noiseAMP = 0.1; % amplitude (re 1) of noise [1]
stimulus = 0; % different stim. types possible (see above)
repeats = 200; % # of averages to do
% [slightly more advanced parameters for stimulus= 1]
modDEPTH = 0.0; % modulation depth in Hz of frequency modulation ** {2}
modFREQ = 5; % appox. freq. of modulation envelope [Hz] ** {40} % (modulation about center frequency)
% --------------------------------
dt = 1/SR; % spacing of time steps
nPts = repeats*Npoints; % total # of points
freq = SR*(0:Npoints/2)/Npoints; % create a freq. array (for FFT bin labeling)
df = SR/Npoints; % quantize the 'stim.' freq. (so to have an integral # of cycles)
fQ = ceil(1000*f/df); % quantized natural freq. [Hz]
[temp indxF] = max(freq == fQ); % find freq. array index for sinusoid
t = (0:1/SR:(nPts-1)/SR); % create an array of time points, Npoints long
% +++
if stimulus==0
% quantized sinusoid + gaussian noise
wQ = cos(2*pi*fQ*t) + noiseAMP*randn(size(t,2),1)';
elseif stimulus == 1
% quantized sinusoid (w/ modulating freq.) + gaussian noise
RND2int=(2*modDEPTH*rand(ceil(modFREQ*max(t)),1))-modDEPTH;
xx=resample(RND2int,ceil(nPts/ceil(modFREQ*max(t))),1);
xx=xx(1:size(t,2));
wQ = cos(t.*(fQ+xx)*'2*pi) + noiseAMP*randn(size(t,2),1)';
end
% +++
% plot (entire) time waveform and associated FFT
if 1==1
figure(1); clf;
subplot(211); plot(t,wQ);
hold on; grid on; xlabel('time [s]'); ylabel('signal'); title('Entire waveform');
subplot(212); plot((SR*(0:numel(wQ)/2)./numel(wQ))/1000,db(abs(rfft(wQ))))
axis([min(freq)/1000 max(freq)/1000 -120 10])
grid on; xlabel('freq. [kHz]'); ylabel('FFT [dB]'); title('FFT (mag.) of entire waveform');
end
% +++
% parse up for time averaging
wAVGtime = zeros(Npoints,1);
for nn=1:repeats
  index = (nn-1)*Npoints + 1;
  wAVGtime = wAVGtime + wQ(index:index+Npoints-1)';
end
wAVGtime = wAVGtime/repeats;
specT = abs(rfft(wAVGtime)); % mag. spec. for time-averaged waveform

[continued]
% parse up for spectral averaging
wAVGspec = zeros(Npoints/2+1,1);
for nn=1:repeats
  clear specTEMP;
  indx = (nn-1)*Npoints + 1;
  specTEMP = abs(rfft(wQ(indx:indx+Npoints-1)));
  wAVGspec = wAVGspec + specTEMP; % Note: this isn’t a waveform, but is a complex spectrum!
end
wAVGspec = wAVGspec/repeats;
specS = wAVGspec; % mag. spec. for spectral-averaged waveform

% plot time-averaged waveform (zoomed in) and FFT
minSPECamp = -70; % dB min for y-axis of spectra
if 1==1
  figure(2); clf;
  subplot(211); plot(t(1:Npoints),wAVGtime);
  hold on; grid on; axis([0 t(Npoints)/40 -1.5 1.5])
  xlabel('time [s]'); ylabel('signal'); title('Temporally averaged waveform (zoomed-in)')
  subplot(212); plot(freq/1000,db(specT)); hold on; grid on;
  plot(freq(indxF)/1000,db(specT(indxF)),'rx','LineWidth',2)
  axis([min(freq)/1000 max(freq)/1000 minSPECamp 5])
  xlabel('freq. [kHz]'); ylabel('FFT [dB]')
  disp(sprintf('Temporal avg. mag. (at stim. freq.)= %g dB',db(specT(indxF))));
end

% plot time-averaged waveform and FFT
if 1==1
  figure(3); clf;
  subplot(211); plot(t(1:Npoints),irfft(wAVGspec)); % convert back to time domain
  hold on; grid on; axis([0 t(Npoints)/40 -1.5 1.5])
  xlabel('time [s]'); ylabel('signal'); title('Spectrally averaged waveform (zoomed-in)')
  subplot(212); plot(freq/1000,db(specS)); hold on; grid on;
  plot(freq(indxF)/1000,db(specS(indxF)),'rx','LineWidth',2)
  axis([min(freq)/1000 max(freq)/1000 minSPECamp 5])
  xlabel('freq. [kHz]'); ylabel('FFT [dB]')
  disp(sprintf('Spectral avg. mag. (at stim. freq.)= %g dB',db(specS(indxF))));
end
SR = 44100; % sample rate {44100}
Npoints = 8192; % length of fft window (# of points)
f = 3.2; % 'stimulus' freq. [kHz]
noiseAMP = 1; % amplitude (re 1) of noise {1}
stimulus = 0; % different stim. types possible (see above)
repeats = 20; % # of averages to do

**Entire waveform**

**FFT of entire waveform**
- Hard to ‘see’ the sinusoid due to the noise
- However the Fourier transform clearly indicates there is a lot of ‘energy’ at the sinusoid’s frequency (i.e., excellent SNR)
So 20 averages seems to allow the sinusoid to at least visibly become apparent and also affects the spectra (e.g., better SNR for temporal averaging, despite a visibly noisier waveform)
Averaging: Spectral vs. Temporal

3.2 kHz sinusoid + Gaussian noise (zero mean, STD=0.5)
SR = 44100 kHz
8192-point window (re FFT)

2 Averages

Temporal Averaging

Spectral Averaging

average in time domain, then take FFT

take FFT, then average magnitudes in spectral domain
Averaging: Spectral vs. Temporal

3.2 kHz sinusoid + Gaussian noise (zero mean, STD=0.5)
SR = 44100 kHz
8192-point window (re FFT)

100 Averages

Temporal Averaging
average in time domain, then take FFT

Spectral Averaging
take FFT, then average magnitudes in spectral domain
Generally, if your response is ‘phase-locked’ to an evoking stimulus, use *temporal averaging* (lower noise floor)

But for measured data are ‘spontaneous’, need to use *spectral averaging*

Note: When spectral averaging, you are effectively throwing out half of your information (i.e., the phase). Hence why it is ultimately inferior....
Ex: Spectral averaging for otoacoustic emissions
Ex: Spectral averaging for otoacoustic emissions

1 (spectral) average

**Human**

**Lizard**
(Anolis carolinensis)

SR = 22050 Hz
8192 point window (re FFT)
Ex: Spectral averaging for otoacoustic emissions

2 (spectral) averages

Human

Lizard

(Anolis carolinensis)

SR = 22050 Hz
8192 point window (re FFT)
Ex: Spectral averaging for otoacoustic emissions

5 (spectral) averages

Human

Lizard (Anolis carolinensis)

Magnitude [dB SPL]

Frequency [kHz]

SR = 22050 Hz
8192 point window (re FFT)
Ex: Spectral averaging for otoacoustic emissions

10 (spectral) averages

Human

Lizard
(Anolis carolinensis)

SR = 22050 Hz
8192 point window (re FFT)
Ex: Spectral averaging for otoacoustic emissions

50 (spectral) averages

Human

Lizard

*(Anolis carolinensis)*

SR = 22050 Hz

8192 point window (re FFT)
Summary (re ‘Data Analysis’)

- It’s useful to keep in mind that DAQ, signal processing, and the notion of ‘data analysis’ (including statistics) are typically done hand-in-hand and thus are closely interrelated.

- Some other useful tips:
  - Keep data files organized! (good lab book notes help enormously too)
  - Take repeated measures (when possible) so to characterize and quantify uncertainty.
  - Don’t be afraid to try different computational approaches to examine with the data. [For example, does converting to the spectral domain help? Any insight gained from a cross-correlation?]
  - How are you going to visualize the data?

→ We’ve dealt with visualizing data indirectly thus far (e.g., regression to determine trends). But there is actually a science to data visualization.....
Post-class exercises

- Fiddle around with the number noise and number of averages in EXaveraging.m to get a feel for the strengths and drawbacks of the two methods.

- Taking the FFT of the entire waveform (as opposed to a shorter segment) in EXaveraging.m appears to lead to a relatively low noise floor. Is an FFT in of itself a form of ‘averaging’?

- Record a waveform 2-3 s long of you whistling at a certain pitch. Divide that waveform up into shorter segments and average both temporally and spectrally. What do you see?

- Create some of the other ‘classic curves’

- Recompute some fractal plots, but instead plot as ‘3D’ curves. Do things look better or worse?

- Write down the ‘principles of graphical excellence’ on paper. Repeat. Repeat. Repeat. Set yourself up in a for loop with N iterations to repeat. [i.e., these are very useful to memorize!]