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
Reminder

\[ x(t) = A_1 \sin(2\pi f_1 t + \phi_1) + A_2 \sin(2\pi f_2 t + \phi_2) \]

→ Changing (relative) phase affects summation

- \( f_1 = 1, f_2 = 1.1 \)
- \( A_1 = 1, A_2 = 1 \)
- \( \phi_1 = 0, \phi_2 = 0 \)

→ Changing (relative) amplitudes affects summation

- \( f_1 = 1, f_2 = 1.1 \)
- \( A_1 = 2, A_2 = 1 \)
- \( \phi_1 = 0, \phi_2 = 0 \)
Speech & Vowels: A E I O U

- Let’s just focus on the initial vowel A (/aI/):

Just a sum of a bunch of sinusoids?
% Example code to just fiddle with basics of discrete FFTs and connections
% back to common real-valued time waveforms
% --> Demonstrates several useful concepts such as 'quantizing' the frequency
% Requires: rfft.m, irfft.m, cycs.m, db.m, cym.m

---

% Stimulus Type Legend
% stimT = 0 - non-quantized sinusoid
% stimT = 1 - quantized sinusoid, one un-quantized sinusoid
% stimT = 2 - one quantized sinusoid, one un-quantized sinusoid
% stimT = 3 - click (i.e., an impulse)
% stimT = 4 - noise (flat, flat spectrum, random phase)
% stimT = 5 - exponentially decaying sinusoid (i.e., NO impulse response)
% stimT = 6 - chirp (flat mag.)
% stimT = 7 - noise (Gaussian, flat spectrum, random phase)
% stimT = 8 - exponentially decaying sinusoid (i.e., NO impulse response)

clear; clf;

% Sample rate
SR = 44100;
Npoints = 8192;

% Frequency for waveforms w/ tones
f = 2580.0;
ratio = 1.22;

% Note: Other stimulus parameters can be changed below

% Compute stimulus
if stimT == 0 % non-quantized sinusoid
    signal = cos(2*pi*f*t);
disp(sprintf(' 
 *Stimulus* - (non-quantized) sinusoid, f = %g Hz 
', f));
elseif stimT == 1 % quantized sinusoid
    signal = cos(2*pi*fQ*t);
disp(sprintf(' 
 *Stimulus* - quantized sinusoid, f = %g Hz 
', f));
disp(sprintf('quantized freq. = %g Hz', fQ));
elseif stimT == 2 % one quantized sinusoid, one un-quantized sinusoid
    signal = cos(2*pi*fQ*t) + cos(2*pi*ratio*fQ*t);
disp(sprintf(' 
 *Stimulus* - two sinusoids (one quantized, one not) 
'));
elseif stimT == 3 % two quantized sinusoids
    signal = cos(2*pi*fQ*t) + cos(2*pi*fQ2*t);
disp(sprintf(' 
 *Stimulus* - two sinusoids (both quantized) 
'));
elseif stimT == 4 % click
    signal = cos(2*pi*fQ*t) + cos(2*pi*fSMP*t);
disp(sprintf(' 
 *Stimulus* - Noise + Sinusoid 
'));
elseif stimT == 5 % noise (flat)
    signal = rand(1,Npoints);
disp(sprintf(' 
 *Stimulus* - Noise 
'));
elseif stimT == 6 % chirp (flat)
    fSMP = 4000.0; % if a chirp (stimT>2) starting freq. (Hz) [freq. swept linearly w/ time]
    fIEQ = 2000.0; % if a chirp (stimT>2) ending freq. (energy usually extends twice this far out)
    fIEQ = cell2mat(df/dt).*df; % quantize the start/end freqs. (necessary?)
    fSMP = cell2mat(df/dt).*df; % LINDAR sweep rate
    fSMP = fIEQ + ((fIEQ-fEQ)/(SR/Npoints)).*t;
    signal = sin(2*pi*fSMP.*t);
disp(sprintf(' 
 *Stimulus* - Chirp 
'));
elseif stimT == 7 % noise (Gaussian)
    Asize = Npoints/2 + 1;
    for n=1:Asize
        theta = rand*2*pi;
        N2(n) = exp(i*theta);
    end
    tNoise = irfft(N2);
    if (abs(min(tNoise)) > max(tNoise))
        tNoise = tNoise/abs(min(tNoise));
    else
        tNoise = tNoise/abs(max(tNoise));
    end
    signal = tNoise;
    disp(sprintf(' 
 *Stimulus* - Noise 
'));
elseif stimT == 8 % exponentially decaying cos
    alpha = 500;
signal = exp(-alpha*t).*sin(2*pi*fQ*t);
disp(sprintf(' 
 *Exponentially decaying (quantized) sinusoid*  
'));
end

% Plot the stimuli as an output sound?
if (1==1), sound(signal,SR); end

% Compute inverse Fourier transform and plot?
if 1==1
    figure(1);
    signalINV = irfft(sigSPEC);
    subplot(211)
    plot(t*1000,signalINV,'r.-','MarkerSize',4)
    legend('Original waveform','Inverse transformed')
end

% Now plot rfft of the signal
% NOTE: rfft just takes 1/2 of fft.m output and normalizes
sigSPEC = rfft(signal);
figure(2); clf; % MAGNITUDE subplot(211)
plot(freq/1000,db(sigSPEC),'ko-','MarkerSize',3)
hold on; grid on;
xlabel('Frequency [kHz]'); ylabel('Phase [cycles]'); grid on;

% Play the stimuli as an output sound?
if (1==1), sound(signal,SR); end

% Compute inverse Fourier transform and plot?
if 1==1
    figure(1);
    sigspecINV = irfft(sigspec);
    fplot(t*10000,signalsINV, 'x', 'MarkerSize', 4)
    legend('Original waveform', 'Inverse transformed')
end
Fourier transforms of basic (1-D) waveforms

stimT = 0 – non-quantized sinusoid

SR = 44100; % sample rate [Hz]
Npoints = 8192; % length of fft window

Note: The phase is ‘unwrapped’ in all the spectral plots

➢ Magnitude shows a peak at the sinusoid’s frequency
Remarkable that the magnitudes are identical (more or less) between two signals with such different properties. The key difference here is the phase: *Timing is a critical piece of the puzzle!***
Fourier transforms of basic (1-D) waveforms

stimT = 6 - chirp (flat mag.)

Hard to see on this timescale, but frequency is changing (increasing) with time
Fourier transforms of basic (1-D) waveforms

stimT = 8 - exponentially decaying sinusoid

This seems to look familiar....
Connection back to.... ... the harmonic oscillator!

- The **steady-state** response of the sinusoidally-driven harmonic oscillator acts like a **band-pass filter**.

- Distinction between **steady-state** response & **impulse response**
  
  [we’ll come back to this]
Exercises

- Write a simple Matlab script that creates two sinusoids, adds them together, then visualizes the resulting waveform. Explore how does the ‘beating pattern’ depends upon the relative frequencies, phases, and/or amplitudes.

- Similar to above, try adding three sinusoids together and observe the variety of effects you can get by changing the relative properties.

- Fiddle around with EXbuildImpulse.m and EXspecREP3.m to get a feel for how different time waveforms have different spectral representations and vice versa

- Using EXbuildImpulse.m, determine the relationship between the impulse, the phase, and the ‘group delay’

- Consider the difference between fft.m & rfft.m, and consider how such relates to the bottom spectral plot shown here
General properties of Fourier transforms

\[
\begin{align*}
f(t) &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} g(\omega) e^{i\omega t} d\omega \\
g(\omega) &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt \\
F(k) &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-ikx} f(x) dx \\
f(x) &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{ikx} F(k) dk .
\end{align*}
\]

Devries (1994)

Kutz (2012)

➢ Don’t be confused by different notations (there are a lot out there!)

➢ Keep in mind that the Fourier transform is defined over the interval \([-\infty, \infty]\), though most ‘signals’ we deal with computationally are finite (e.g., \([-L, L]\)) (the resolution here is an implicit assumption of a ‘periodic boundary condition’; we’ll come back to this)

➢ Connection back to previous topics covered:

Further, the kernel of the transform, \(\exp(\pm ikx)\), describes oscillatory behavior. Thus the Fourier transform is essentially an eigenfunction expansion over all continuous wavenumbers \(k\). And once we are on a finite domain \(x \in [-L, L]\), the continuous eigenfunction expansion becomes a discrete sum of eigenfunctions and associated wavenumbers (eigenvalues).
General properties of Fourier transforms

- The Fourier transform is a linear process

That is, if $f_1(t)$ and $f_2(t)$ are two functions having Fourier transforms $g_1(\omega)$ and $g_2(\omega)$, then the Fourier transform of $f_1(t) + f_2(t)$ is

$$g(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} [f_1(t) + f_2(t)] e^{-i\omega t} \, dt$$

$$= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f_1(t) e^{-i\omega t} \, dt + \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f_2(t) e^{-i\omega t} \, dt$$

$$= g_1(\omega) + g_2(\omega).$$

→ Linearity is a seemingly innocuous property that has vast implications... (we’ll see some in later lectures)

- Another key property is that of scaling:

$$\mathcal{F}[f(\alpha t)] = \frac{1}{|\alpha|} g\left(\frac{\omega}{\alpha}\right)$$

$$\mathcal{F}^{-1}[g(\beta \omega)] = \frac{1}{|\beta|} f\left(\frac{t}{\beta}\right)$$

Devries (1994)
General properties of Fourier transforms

\[ f(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} g(\omega)e^{i\omega t}d\omega \quad \mathcal{F}[f(\alpha t)] = \frac{1}{|\alpha|} g\left(\frac{\omega}{\alpha}\right) \]

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

- Due to this scaling, as \( f(t) \) gets narrower (i.e., more localized in time), \( g(\omega) \) gets broader (i.e., more spread out across frequency).

In large part, the role of the additional functions is to cancel the oscillations. Thus, the more localized a function is in time, the more delocalized it is in frequency.

This is more than a casual observation — it’s a fundamental property of Fourier transforms, and has direct physical consequences. Usually stated in terms of position and momentum rather than time and frequency, the statement is that the product of the width of the function, \( \Delta x \), and the width of the transform of the function, \( \Delta p \), is always greater than or equal to a specific nonzero value, \( \hbar \) — Heisenberg’s uncertainty principle.

Devries (1994)
General properties of Fourier transforms

A handful of other properties arise (e.g., shifting, time reversal), which give rise to numerous symmetries that can be summarized as follows:

- If \( f(t) \) is real,
  - if \( f(t) \) is imaginary,
  - if \( f(t) \) is even,
  - if \( f(t) \) is odd,
  - if \( f(t) \) is real and even,
  - if \( f(t) \) is real and odd,
  - if \( f(t) \) is imaginary and even,
  - if \( f(t) \) is imaginary and odd,

  then \( \Re g(\omega) \) is even and \( \Im g(\omega) \) is odd;
  then \( \Re g(\omega) \) is odd and \( \Im g(\omega) \) is even;
  then \( g(\omega) \) is even;
  then \( g(\omega) \) is odd;
  then \( g(\omega) \) is real and even;
  then \( g(\omega) \) is imaginary and odd;
  then \( g(\omega) \) is imaginary and even;
  then \( g(\omega) \) is real and odd.

Another key feature is that of derivatives:

\[
\mathcal{F}[f'(t)] = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f'(t) e^{-i\omega t} \, dt
\]

Integrating by parts has:

\[
\mathcal{F}[f'(t)] = \frac{e^{-i\omega t}}{\sqrt{2\pi}} f(t)|_{-\infty}^{\infty} + \frac{i\omega}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{-i\omega t} \, dt
\]

\[
\mathcal{F}[f'(t)] = i\omega g(\omega)
\]

→ The first term on the right-side must be zero

Devries (1994)
Aside: Using Fourier transforms to solve linear differential equations

\[
\hat{f}'(n) = (ik)^n \hat{f}.
\]

Note: This topic is a bit beyond the scope of 2030, but is worth pointing out here for future reference

\[
\mathcal{F}[f'(t)] = i\omega g(\omega)
\]

Basic idea is generalizable to higher order derivatives

Consider the linear, non-autonomous ODE:

\[
y'' - \omega^2 y = -f(x) \quad x \in [-\infty, \infty]
\]

Take Fourier transform of both sides and work through:

\[
\hat{y}'' - \omega^2 \hat{y} = -\hat{f}
\]

\[
-k^2 \hat{y} - \omega^2 \hat{y} = -\hat{f}
\]

\[
(k^2 + \omega^2)\hat{y} = \hat{f}
\]

\[
\hat{y} = \frac{\hat{f}}{k^2 + \omega^2}
\]

\[
y(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{ikx} \frac{\hat{f}}{k^2 + \omega^2} dk
\]

We end up with an integral solution that can then either be evaluated analytically or numerically

→ Other avenues deal with PDEs and integrals, with many practical implications in physics (e.g., delta functions in quantum mechanics, Parseval's identity), though such is beyond our scope here

Kutz (2012)
Aside: Using Fourier transforms to solve linear differential equations

Inner ear model consists of coupled (linear) harmonic oscillators

Basic equations of motion

\[ I \ddot{\theta} + R \dot{\theta} + K \theta + h \sum_{n=1}^{N} k_n (\xi_y - \xi_n) \equiv rA(p_v - p_t) \]

\[ m_n \ddot{\xi}_n + r_n \dot{\xi}_n + k_n (\xi_n - \xi_y) = 0. \]

Exploiting the assumed linearity, we adopt harmonic time dependence and represent dynamical variables by Fourier coefficients (uppercase) at angular frequency \( \omega = 2\pi f \) [i.e., \( \xi_y(t) = \Xi_y(\omega) e^{i\omega t} \)]. Equation (2) then simplifies to

\[ \Xi_n = \frac{\Xi_y}{1 - \beta_n^2 + i\beta_n/Q_n}, \]

No need to numerically integrate in order to get (steady-state) frequency response

\[ \Xi_y \left[ (i\omega)^2 I + i\omega R + K + h^2 \sum_{n=1}^{N} k_n \frac{-\beta_n^2 + i\beta_n/Q_n}{1 - \beta_n^2 + i\beta_n/Q_n} \right] \]

\[ = rhA(p_v - p_t), \]

Bergevin & Shera (2010)
Reminder

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)
Discrete Fourier transforms

- Many signals we deal with computationally are ‘digital’. That is, the signal is discretely sampled at intervals $\Delta t$ at a Sample Rate (SR, $1/\Delta t$) [Hz].

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

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

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.
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:

$$f(t) = \sum_{n=-\infty}^{\infty} c_n e^{i2\pi nt/T}$$

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

- 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:

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

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

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

Devries (1994)
A means to efficiently compute a DFT

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 $2, 4, 8, 16, 32, 64, 128, 256, \ldots$.
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.
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*

  ![Fourier Transform Symmetry](image)

  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

  ![Square Wave](image)

  ![Frequency Spectrum](image)

  ➔ *rfft.m* does this, as well as normalizes the amplitudes

  \[
  \mathcal{F}[f(\alpha t)] = \frac{1}{|\alpha|} g\left(\frac{\omega}{\alpha}\right)
  \]

  [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

```matlab
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;
```

\[ \Delta \omega = \frac{2\pi}{T} \]
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

22.7 µs intervals
8192 samples

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!

- *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)
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.

![Image of three 8-point Hann window functions](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
Comparison of start of interval of quantized vs non-quantized sinusoids

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]