Chapter 11: The discrete time Fourier transform, the FFT, and the convolution theorem

Joseph Fourier 1768-1830
The discrete time Fourier transform

• The main idea: A periodic signal can be expressed as the sum of sine and cosine waves of different amplitudes and frequencies.

• A periodic signal is a signal which repeats over uniformly spaced intervals of time, from infinite past to infinite future...
The discrete time Fourier transform


The discrete time Fourier transform
The discrete time Fourier transform

The importance of the Fourier transform

• This information is an integral part of EEG processing – this is how we get the different frequencies from the signal!

• This allows us to look at the amplitude (power), or phase of different frequencies across time, and or space (channels).
Making waves

%the sine wave formula: Amplitude * sin(2pi*frequency*time + phase angel)

- srate = 1000; % sampling rate of 1 kHz
- time  = -1:1/srate:1;
- freq  = 10; % in Hz
- amp   = 2; % amplitude, or height of the sine wave
- sine_wave = amp.*sin(2*pi*freq.*time); % Note that you need the .* for point-wise vector multiplication.
- figure
- plot(time,sine_wave)
- set(gca,'ylim',[-3 3]) % this adjusts the y-axis limits for visibility
- title('My first sine wave!')

% now try making a sine wave using a different frequency and amplitude:
- srate = 1000; % sampling rate of 1 kHz
- time  = -1:1/srate:1;
- freq  = 5; % in Hz
- amp   = 1; % amplitude, or height of the sine wave
- sine_wave = amp.*sin(2*pi*freq.*time);
- hold on
- plot(time,sine_wave,'-r')
- set(gca,'ylim',[-3 3]) % this adjusts the y-axis limits for visibility
- title('My second sine wave!')
summing up the waves

Figure 11.2
The discrete time Fourier transform

%% Figure 11.4
time=-1:1/srate:1;
% create three sine waves
s1 = sin(2*pi*3*time);
s2 = 0.5*sin(2*pi*8*time);
s3 = s1+s2;
% plot the sine waves
figure
for i=1:3
    subplot(2,3,i)  % plot sine waves, using the eval command
    eval(['plot(time,s' num2str(i) ')' ]);
    set(gca,'ylim',[-1.5:.5:1.5])
% plot powersubplot(2,3,i+3)
    f = eval(['fft(s' num2str(i) ')/length(time)' ]));
    hz = linspace(0,srate/2,(length(time)-1)/2);
    bar(hz,abs(f(1:(length(time)-1)/2))*2)
    set(gca,'xlim',[0 11],'xtick',0:10,'ylim',[0 1.2])
end
The discrete time Fourier transform in 3D

Figure 11.5
The discrete time Fourier transform

- The highest frequency (or number of frequencies that could be represented) is \( N \) (number of data points) / 2 + 1
- Why? To avoid aliasing the signal.
- **Nyquist’s theorem**: A sample rate twice as fast as the highest signal frequency will capture that signal perfectly
- This means that the highest frequency which can be accurately represented is half of the sampling rate
- Solution: sample fast enough

- The lowest frequency that can be represented is 1/T
- Why? Because the (periodic) signal has to complete at least one full cycle
- Solution: sample long enough epochs.
- This would increase the resolution (which is 1/T). For example, a 10 seconds epoch would give a frequency resolution of 1/10 Hertz
The discrete time Fourier transform

%%% back to Figure 11.5

N       = 10;         % length of sequence
data    = randn(1,N); % random numbers
srate   = 200;        % sampling rate in Hz
nyquist = srate/2;    % Nyquist frequency -- the highest frequency you can measure in the data

% initialize Fourier output matrix
fourier=zeros(size(data));

% These are the actual frequencies in Hz that will be returned by the
% Fourier transform. The number of unique frequencies we can measure is
% exactly 1/2 of the number of data points in the time series (plus DC).
frequencies = linspace(0,nyquist,N/2+1);
time = ((1:N)-1)/N;

% Fourier transform is dot-product between sine wave and data at each frequency
for fi=1:N
    sine_wave   = exp(-1i*2*pi*(fi-1).*time);
    fourier(fi) = sum(sine_wave.*data);
end
fourier=fourier/N;
The inverse Fast Fourier transform

Figure 11.7
Discrete time Fourier Transform vs. Fast Fourier Transform

- Real, imaginary and complex numbers?
- Chapter 13?
- Doc?
- John?

**Bottom line:** The FFT takes less time! (and uses powers of 2)
How much? “taking $N = 2^{10}$ and calculate the time, the DFT takes 1247 years while the FFT takes only 10.67 min.”
Stationary = periodic?

Figure 11.9
Zero padding

Zero padding does not increase the amount of data we have (frequencies) but it improves our frequency resolution – our ability to distinguish between adjacent frequencies. Matlab automatically adds zeroes when fft(a number larger than N)

No zero padding

A lot of zero padding

And again we are promised more information in chapter 13...
The convolution theorem

- Convolution in the time domain = multiplication in the frequency domain

F denotes the Fourier transform operator, so F{f} and F{g} are the Fourier transforms of f and g.

\[ F\{f\ast g\} = F\{f\} \times F\{g\} \]
\[ F\{f \times g\} = F\{f\} * F\{g\} \]
\[ f \ast g = F^{-1}[F\{f\} \times F\{g\}] \]

And there's also proofs ....

Bottom line: time domain convolution is slow and frequency domain convolution is fast.
The convolution theorem

Figure 11.11
The convolution theorem

**Figure 11.12**
End of presentation

• Thank you!

• Good night...