MATLAB for Audio Signal Processing

P. Professorson
UT Arlington Night School

Getting data into the computer

MATLAB for Audio Signal Processing

- Getting real world data into your computer
- Analysis based on frequency content
  - Fourier analysis
- Modifying the spectrum
  - Filtering signal bands
    - FIR filters
    - IIR filters
- Other stuff

Aliasing

- Sampling frequency must be high enough to characterize the signal correctly.
  - Fs \( \geq 2 \) fmax
- For a given Fs, signal components higher than Fs/2 (Nyquist frequency) are forcibly removed using an anti-aliasing filter prior to sampling.
Audio signals

- Audio spectrum for music lies between 20Hz to 20KHz
- This gives a Nyquist sampling rate of 40KHz
- CD audio uses a sampling rate of 44.1 KHz
- 16 bits per sample
- WAV files store uncompressed samples
- MP3 files store compressed data

Importing audio in MATLAB

- WAV files:
  - `[data sr nbits] = wavread('song.wav');`
  - `sr`: sampling rate
  - `nbits`: number of bits per sample
  - For mono audio, data is a vector containing audio samples
  - For stereo audio, data is a 2-D vector
- MP3 files – convert them to WAV format using an external MP3 decoder, then use wavread

Playing back your audio

- Create an audioplayer object
  - `ap = audioplayer(data, sr);`
- Play the audio file using the object just created
  - `play(ap);`
    - This plays the audio in the background while the rest of your script continues working.
    - `isplaying(ap)` returns 1 if the file is still playing, and 0 if the file has finished playback.
  - `playblocking(ap);`
    - This halts execution till the audio clip stops playing
Audio spectrum

- Human ear's response is non-linear. It is good at discriminating low frequencies, but not so good at distinguishing between higher frequencies.
- Spectrum analyzers on music players have logarithmic frequency scales.
- Octave is a frequency scale that doubles at every level.
  - \( F_1 = 30 \text{Hz}, \ F_2 = 60 \text{Hz}, \ F_3 = 120 \text{Hz} \ldots \)

Discrete-time Fourier Analysis

- Discrete Fourier Transform gives a discrete-frequency representation of a discrete time signal.
- An N-point DFT of a signal will produce N numbers, each representing the signal content at uniformly spaced frequencies between \(-Fs/2\) to \(+Fs/2\) (not in that order).
- Fast Fourier Transform (FFT) is a (class of) algorithm(s) that produce the DFT efficiently - \(O(N\log(N))\) vs \(O(N^2)\) for regular DFT.

MATLAB's FFT

- \( f = \text{fft}(\text{signal}, \ N); \)
  - Produces N-point DFT of a signal
  - \( N \) should be > length of signal to preserve fidelity, preferably a power of 2 for efficiency
- For a real signal, \( f \) is generally complex. Its absolute value is used to specify its strength.
- \( \text{signal2} = \text{ifft}(f, \ M) \)
  - Reproduces a signal of length \( M \) whose N-point DFT is specified in \( f \)

FFT fun facts

- \( f = \text{fft}(\text{signal}, \ N); \)
  - \( f(1) \) is the DC component / signal average
  - Useless with respect to audio spectrum
  - \( f(2) \) is the strength at frequency \( Fs/N \)
  - \( f(k) \) is the strength at frequency \((k-1)Fs/N\)
  - \( f(N/2+1) \) is the strength at Nyquist frequency \( Fs/2 \)
  - Useless as it varies with phase of the signal
  - Beyond this are negative frequencies which are mirror images of positive frequencies (for real signals)
  - \( f \) needs to be scaled as \( f/2/N \)
  - FFT bins represent the exact frequency. Frequencies lying between bins are not accurately represented.
- Signal power spectrum in decibels is given by
  - \( 20\log_{10}(\text{abs}(f)); \)

http://www.zytrax.com/tech/audio/equalization.html
**Direct FFT spectrum modification**

- We can modify the shape of the spectrum ‘f’ before we invert it.
- Need to avoid abrupt changes in bands.
- \( f_{\text{new}} = f \cdot \text{desired}_\text{gains} \);
- \( \text{signal}_{\text{new}} = \text{ifft}(f_{\text{new}}, N1) \);
  - \( N1 \) is the length of the original signal

**Disadvantages of the FFT method**

- FFT is good for analysis of the signal, but not good for changing the signal spectrum because:
- FFT assumes that the signal is periodic, i.e. the signal supplied to it repeats itself.
  - This is not true in real life
  - We need to apply a window function so that the signal decays to zero at its both ends.
- FFT order needs to be greater than the signal's length to allow faithful reproduction using inverse FFT.

**Band-pass filter banks**

- Audio spectrum is divided into non-uniformly spaced bands – the ones you see on an equalizer.
- Audio is passed through each filter, and the signal content at the output gives the content in that frequency band.
  
  *Filter 1: 22 Hz – 44 Hz, Filter 2: 44 Hz – 88 Hz
  Filter 3: 88 Hz – 176 Hz, ...
  Filter 9: 5.5 KHz – 11.5 KHz, Filter 10: 11.5 KHz – 22 KHz*

**Time-domain filters**

- Finite Impulse Response (FIR) filters
- Infinite Impulse Response (IIR) filters

- What is impulse response?

  ![Diagram](image)

  This relationship can be expressed as \( y(n) = x(n) * h(n) \)
  
  Where * is the convolution operation, and \( h(n) \) is called the system's impulse response.

  Most operations like frequency selective filtering, averaging, etc. are linear.
  Non-linear operations such as median filtering cannot be performed with convolution.
FIR filters

- The impulse response \( h(n) \) is finite length
- Can be implemented as →
- Impulse response has \( N \) coefficients \( h(1) \) to \( h(N) \), also referred to as \( b_0, b_1, \ldots, b_{N-1} \) as shown in the fig.
- \( N \) is called the filter order
- We need access to \( N \) past values of \( x \) to perform this operation. A circular buffer of size \( N \) is used to store these in a realtime system.

FIR design steps

- Determine the desired frequency response
- Find \( h(n) \) using inverse Fourier transform
- This \( h(n) \) has infinite elements.

- The final step is to shift the whole filter so that there are no samples in negative time.
- The negative time samples imply that the filter needs to access future \( N/2 \) values of \( x \) ahead of current time \( n \). This results in a filter delay, as we need to wait for the next \( N/2 \) samples to arrive before producing a valid output.

- Truncate it to \( N \) elements using a window function.
- Using an abrupt rectangular window will cause ringing artifacts. (Gibb's phenomenon)
- A smoothing window such as a Hamming window is used.
  \[
  h = h^*
  \]
**FIR filters in MATLAB**

- `b = fir1(N, wcn)` produces an FIR filter with `N+1` coefficients.
  - `wcn` is the normalized cutoff frequency
  - For a low pass filter with cutoff `fc` Hz, `wcn = fc/fNyquist`.
  - For a band pass filter with cutoffs `fcl` and `fch`, `wcn = [fcl fch]/fNyquist`.
  - Remember that `fNyquist = Fs/2`.
- To visualize this filter, use the filter visualization tool:
  - `fvtool(b);`

**IIR Filters**

- Structure is very similar to FIR filters except that output `y` is also fed back through weights `a_n`.
- We need to store `N` previous values of `x` and `y`.
- The required filter order `N` is much smaller than FIR filters for the same performance.

**How MATLAB will do it**

- Given your filter specs, determine whether you need a Butterworth filter, Chebychev-I/II filter, or an Elliptical filter.
- Look up the corresponding command from documentation. Each command has a number of options to specify your filter specs.
- `[b a] = butter(N, wc);` will produce a Butterworth filter of order `N`, with normalized cutoff frequency(ies) given in `wc`. The filter coefficient are conveniently returned.
  - For Band Pass response, the filter order is `2*N`, and `wc = [fc1 fc2]/fNyquist`.
Applying the IIR filter to a signal

- \( y = \text{filter}(b, a, x); \) will apply the IIR filter represented by arrays \( b \) and \( a \) to the signal \( x \), producing the filtered output \( y \).
- If \( x \) is windowed, the filter's final conditions must be passed as initial conditions when filtering the next window.
- To visualize this filter, use the filter visualization tool:
  
  - \( \text{fvtool}(b, a); \)

FIR vs IIR

- Both have their merits and applications
- FIR filters produce a linear phase response, while IIR filters severely distort the phase.
- FIR filters need a much higher order than an IIR filter to obtain the same quality of response. Consequently the filter delay is considerably large for FIR filters.
- FIR filters are inherently stable, while poorly designed IIR filters may over/underflow, and IIR filters are also affected by precision a great deal.

Web References

- MATLAB MP3 reading functions: http://labrosa.ee.columbia.edu/matlab/mp3read.html
- Octaves frequency scale:
  - http://www.recordingeq.com/EQ/eq0400/OctaveEQ.htm
- Spectrum Analysis Windows:
  - https://ccrma.stanford.edu/~jos/sasp/Spectrum_Analysis_Windows.html
- Spectrum Analyzer Implementation on DSP:
- FFT Spectrum Analyzers:
- FIR filters:
- IIR filters:
- Equalization and other DSP implementations: