Lecture 6: Modifying Sampled Signals

The Digital World of Multimedia
Prof. Mari Ostendorf
Questions

■ Do you want solutions posted for Lab 1?
  □ yes

■ Extra lab hours – what times are good?
  (We have an undergrad assistant starting next week.)
  □ Weds after class?? (tbd)

■ My office hours:
  □ Friday change to 9-11
  □ Extra office hours – Thurs after 3:30 (tbd)
Goals for Today

- Working with digital signals in MATLAB
- Simple methods for modifying sounds
- Filtering
- Signal generation
Creating a Signal in MATLAB

- Desired signal: $x(t) = \cos(2\pi F_0 t)$, $F_0 = 100\text{Hz}$
- Pick a sampling rate, say $F_s = 250\text{Hz}$ ($F_s = 250; T_s = 1/F_s$;)
  - Corresponds to $T_s = 1/250 \text{ sec}$
- To make a 3 sec signal:
  - Create a time vector: $t_3 = [0:T_s:3]$;
  - Use that in the cosine function to get a value for $x(n)$ at each time $nT_s$:
    - In math: $x(n) = \cos(2\pi F_0 n T_s)$
    - In MATLAB: $x = \cos(2\pi F_0 t_3)$;
- To play the signal, use:
  - `sound(x,Fs)` NOT `sound(x)`.
    - Why?
      - Need $F_s$ to get back to the analog world. If not specified, $F_s = 8192\text{Hz}$ is used.
Time-domain view of signal recovery

- Straight line interpolation is like putting a triangle at each sample and adding them up.
- Using a smoother function works better (the picture is an approximation).
- In the frequency domain, it is an ideal low pass filter.
Reading a Signal into MATLAB

Demo: What’s wrong with this?
- haha = wavread('haha2');
- sound(haha);

Right approach(es)…
- [haha, Fs, bits] = wavread('haha2');
- or [haha, Fs] = wavread('haha2');
- sound(haha,Fs);
Modifying Sounds

- **Amplitude operations**
  - Multiplying signals (e.g. for fading)
  - Adding two signals (mixing)

- **Time operations**
  - Time reverse
  - Time delay
  - Time-scale modification (speed up, slow down)

- **Changing frequency content: filtering**
Simple Signal Operations

- **Amplitude scaling:**
  
  \[ y(n) = 3x(n) \]

  Changes loudness, especially useful if you filter out frequency content and end up with a lower energy signal (watch out for clipping)

- **Mixing signals: addition** (optionally with scaling)
  
  \[ y(n) = x_1(n) + 0.2x_2(n) \]

- **Changing the envelope, as in fading:**
  
  multiplication

  \[ y(n) = x_1(n)x_2(n) \] (use y=x1.*x2 in MATLAB)
Simple Signal Operations

- Time reverse (flipud(x) or fliplr(x) in MATLAB)
  - Examples: music 🎵 🎵 speech 🎵 🎵

- Time Delay: \( y(n) = x(n-T_0) \)
  - in MATLAB: delayed_x = [zeros(T0,1);x];

- Echo: (Multiple time delays added)
  \( y(n) = x(n) + 0.5 \times x(n-T_0) + 0.25 \times x(n-2T_0) \)
  - MATLAB example: relax 🎵
    
    ```matlab
    >> pause=zeros(1500,1);
    >> echo=[relax;pause;relax*.4;pause;relax*.16;pause;relax*.05];
    >> sound(echo) 🎵
    ```
Time Scaling

Drop samples:
- speed up in time
- higher pitch
- higher frequency content
  (watch out for aliasing!)

Insert samples:
- slow down in time
- lower pitch
- lower frequency content
  (no aliasing worries)

2X: Drop in MATLAB:
```matlab
for i=1:len/2
    cos2(i)=cos1(2*i-1);
end
```

0.5X: Insert in MATLAB:
```matlab
for i=1:len-1
    cos3(2*i-1)=cos1(i);
    cos3(2*i)=0.5*(cos1(i)+cos(i+1));
end
```
Filtering: Changing Frequency Content

- **Low pass filter:**
  - Keeps low frequencies, suppresses high frequencies
  - Smoothing effect

- **High pass filter:**
  - Keeps high frequencies, suppresses low frequencies
  - Emphasizes abrupt changes

- **Band pass filter:**
  - Keeps some mid range of frequencies, suppresses both high and low frequencies
Speech Example

Original Signal

Low Pass Filtered

High Pass Filtered
Example: Castanets & Guitar
Square Wave Example
Different Varieties of Filters

Different LPFs
All with cut-off at 100Hz
More on Filtering

- Can also be used to “shape” frequency content
- Any function of the form
  \[ y(n) = a_1 y(n-1) + a_2 y(n-2) + \ldots + a_p y(n-p) \]
  \[ + b_0 x(n) + b_1 x(n-1) + \ldots + b_m (n-m) \]
  can be used to shape/change frequency content (input is \( x(n) \), output is \( y(n) \))
- Leads us to the idea of generating signals
For the ambitious...
More on Time Scaling

What can you to avoid aliasing when increasing pitch?
- Low pass filter (LPF) the sound before you time scale
- Example:
  - Fs=16kHz, signal uses full 8kHz band,
  - want to time scale by a factor of 2
  - LPF at 4kHz cutoff (0.5 cutoff in digital filter design)

Can you change the pitch without changing the duration?
- Yes, but it requires tricky cutting/pasting of pieces of the signal (beyond the scope of this class)
Resampling vs. Time Scaling

- Playing a signal with a different Fs has the same effect as time scaling, BUT
  - It’s not a solution if you want to do multiple time scale changes within one sound file
  - Time scaling: add/drop samples, keep Fs the same

- Resampling: changing Fs but *NOT* changing the duration/pitch of the signal
  - Needed for combining signals of different sampling rates, or for taking up less disk space
  - Resampling: add/drop samples, change Fs
  - When decreasing Fs, watch out for aliasing!
  - Handy function for resampling….
Filtering in MATLAB

You can implement a filter in MATLAB by:

- writing a for loop to implement the system function (as in Lab2 “filtersounds.m”)
- Calling the built-in “filter” function, as in:
  \[ y = \text{filter}(b, a, x) \] where \( b = [b_0, b_1, \ldots b_m] \), \( a = [1, a_1, a_2, \ldots a_p] \)

You can design a filter...

- By hand, as in: \( a = 1; b = [1, 1, 1]/3; \)
- Using filter design commands, such as:
  \[ [b \ a] = \text{butter}(n, Wn, 'ftype') \] OR \( b = \text{fir1}(n, Wn, 'ftype'); a = 1; \)

- cutoff(s): Ex: \( Wn = 0.5 = 4k/8k \) For 4k analog cutoff of signal with 8k range (Fs=16k)

order (size of \( a/b \))

low, high, or stop