MUS421 Lecture 8A
FFT Signal Processing: The Overlap-Add (OLA) Method for Fourier Analysis, Modification, and Resynthesis

Julius O. Smith III (jos@ccrma.stanford.edu)
Center for Computer Research in Music and Acoustics (CCRMA)
Department of Music, Stanford University
Stanford, California 94305

June 27, 2020

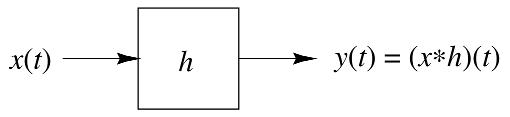
Outline

- Convolution of Short Signals (one window)
- Convolution with Long Signals (multiple windows)
- Time Varying FIR Filtering (new each window)
- Poisson Summation Formula (window COLA test)

Course Outline Up to Now

- Fourier Theorems
- FFT Windows
- FIR Filter Design
- Tonal Spectrum Analysis
- Noise Spectrum Analysis
- Audio Spectral Display (Spectrograms)
- FFT Signal Processors
  [we are here]

Frequency-Domain Convolution



In discrete time,

$$y(n) \stackrel{\Delta}{=} (x{*}h)(n) \stackrel{\Delta}{=} \sum_{m=0}^{\infty} x(m)h(n{-}m), \quad n = 0, 1, 2, \ldots$$

where x and h are assumed causal

Convolution Theorem:

$$(h * x) \longleftrightarrow H \cdot X$$

or

$$\text{DTFT}_\omega(h * x) = H(\omega)X(\omega)$$

where H and X are the DTFTs of h and x, respectively.

FFT Convolution

If x and h have *finite (nonzero) support*, then so does $x * h$, and we may *sample* the frequency axis of the DTFT:

$$\text{DFT}_k(h * x) \;=\; H(\omega_k)X(\omega_k)$$

where H and X are the N-point DFTs of h and x, respectively.

The DFT performs *circular* (*cyclic*) convolution:

$$y(n) \stackrel{\Delta}{=} (x * h)(n) \stackrel{\Delta}{=} \sum_{m=0}^{N-1} x(m)h(n-m)_N$$

where $(n-m)_N$ means "$(n-m)$ modulo N"

Two methods:

- direct calculation of the summation $= \mathcal{O}(N^2)$
- frequency-domain approach $= \mathcal{O}(N \lg N)$
  - DFT both x and h to obtain X and H
  - Multiply pointwise to obtain $Y = X \cdot H$
  - Inverse DFT to get y in the time domain

Audio FIR Filters

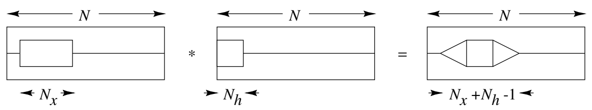
Do we need FFT convolution for practical audio filters?

Yes:

- FFT convolution [$\mathcal{O}(N \lg N)$] starts beating time-domain convolution [$\mathcal{O}(N^2)$] for $N \geq 128$ or so (on a single CPU)
- The nominal "integration time" of the ear, defined, *e.g.*, as the reciprocal of a Bark critical-bandwidth of hearing, is greater than 10ms below 500 Hz
- At a 50 kHz sampling rate, this is 500 samples
- FIR filters shorter than the ear's "integration time" can generally be characterized by their magnitude frequency response (no perceivable "delay effects")
- Thus, even "perceptually instantaneous" FIR filters can easily be *hundreds of taps long*
- For longer FIR filters, the FFT advantage is that much greater
- We conclude that FFT convolution is an important implementation tool for FIR filters in digital audio

Zero Padding for Acyclic FFT Convolution

Recall: *Zero-padding* embeds *acyclic* convolution in *cyclic* convolution:



- In general, the nonzero length of $y = h * x$ is $N_y = N_x + N_h - 1$
- Therefore, we need FFT length $N \geq N_x + N_h - 1$ (zero-padding factor $\stackrel{\Delta}{=} N/N_x$)
- When zero-padding is insufficient ($N < N_x + N_h - 1$), convolution terms "wrap around" in time (due to modulo indexing), giving *time aliasing*
- We typically zero-pad even more (to the next power of 2) so we can use the split-radix Cooley-Tukey FFT for maximum speed

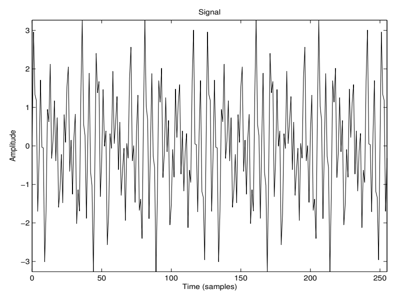
FFT Convolution Example 1: Low Pass Filtering

Problem Statement: Design and implement a low-pass filter

- $F_{cut} = 600$Hz
- Filter length $L = 257$ taps
- Signal $x(n)$ is a sum of sinusoidal components (440, 880, 1000, 2000Hz)
- We can choose frame length $M = 256$ so that the FFT size $N = M + L - 1 = 512$ has just enough zero-padding for frequency-domain filtering without aliasing in the time domain

Example 1: Input Signal

```
% Signal parameters:
f = [ 440 880 1000 2000 ];      % frequencies
M = 256;                        % signal length
Fs = 5000;                      % sampling rate

% Generate a signal by adding up sinusoids:
x = zeros(1,M); % pre-allocate 'accumulator'
n = 0:(M-1);    % discrete-time grid
for fk = f;
    x = x + sin(2*pi*n*fk/Fs);
end
```

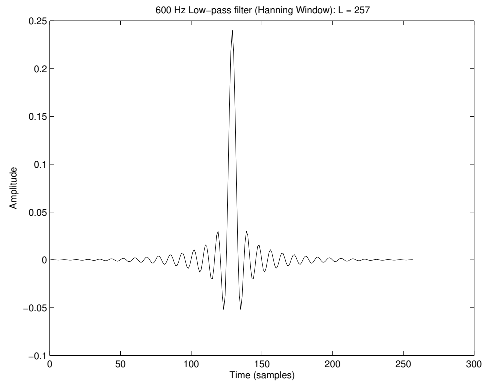


Input Signal: Time-Domain Waveform

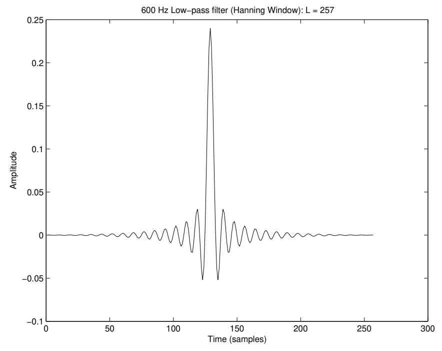
Example 1: FIR Filter Design

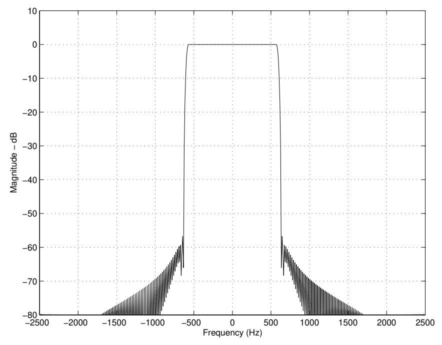
```
% Filter parameters:
L = 257;    % filter length
fc = 600;   % cutoff frequency

% Design the filter using the window method:
hsupp = (-(L-1)/2:(L-1)/2);
hideal = (2*fc/Fs)*sinc(2*fc*hsupp/Fs);
h = hamming(L)' .* hideal; % h is our filter
```

Impulse Response h



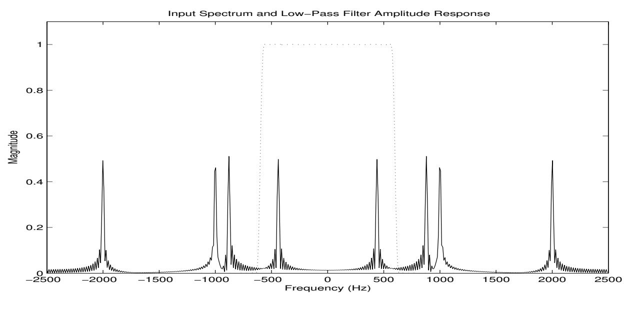
Impulse Response and Amplitude Response





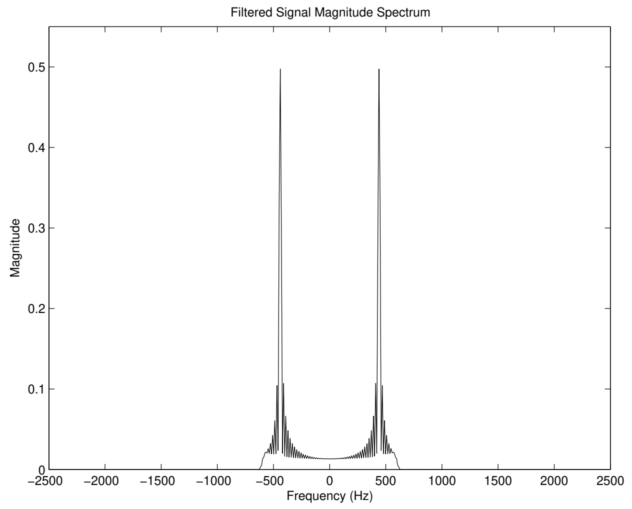
Prepare for Frequency-Domain Convolution

```
% Choose the next power of 2 greater than L+M-1
Nfft = 2^(ceil(log2(L+M-1))); % or 2^nextpow2(L+M-1)

% Zero pad the signal and impulse response:
sigzp = [ sig zeros(1,Nfft-M) ];
hzp = [ h zeros(1,Nfft-L) ];

% Transform the signal and the filter:
S = fft(sigzp);
H = fft(hzp);
```



Perform Frequency-Domain Convolution

```
Sfilt = S .* H; % Filter = frequency-domain window
```

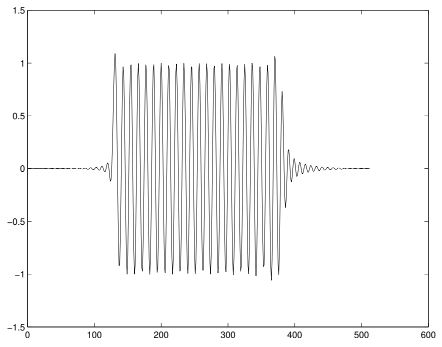


The filter output is now obtained by the inverse Fourier transform

Return to Time Domain

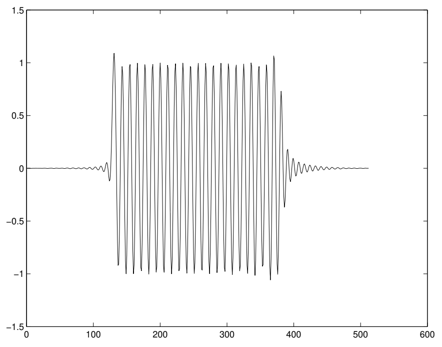
After IFFT, imaginary part is not quite zero due to finite numerical precision:

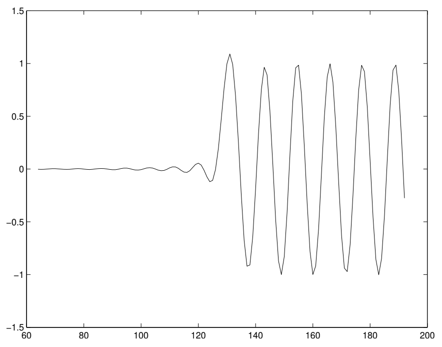
```
sfilt = ifft(Sfilt);
rmserrpct = 100*norm(imag(sfilt))/norm(sfilt) % check
sfilt = real(sfilt);
```



- For a signal of length ≈ 4000, this was 2-3 times faster than `conv()` in Matlab
- Note approximately equal amounts of "pre-ringing" and "post-ringing" due to filter being linear phase (symmetry would be exact if signal were left-right symmetric)

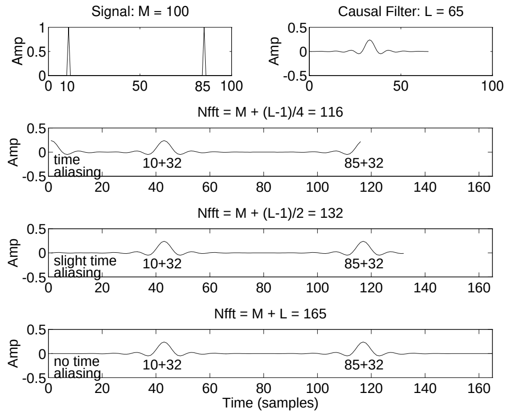
Zoom-In on Start-Up Transient





Example 2: Time Domain Aliasing

This example shows the effect of insufficient zero padding (undersampling in the frequency domain)



- Lowpass filter length $L = 65$
- Data frame length $M = 100$ (impulse at time 0 and $M - (L-1)/4 = 85$)
- $N_{\text{fft}} = M + (L-1)/4 = 116, M + (L-1)/2 = 132, M + L = 165$
- $N_{\text{fft}} \geq M + L - 1$ avoids time-domain aliasing

FFT Convolution Across Frames

For FIR-filtering of very long signals, the *OverLap-Add* (OLA) method is often used:

- Partition the input signal into adjacent *frames* (*i.e.*, using the "rectangular window")
- Process each frame as in the previous example (zero pad, FFT, window by H, IFFT)
- *Sum* the (overlapping) frames into an *overlap-add output buffer*

Extensions:

- The FIR filter h can also be broken up into frames
  - Basis for *low-latency FFT convolution* techniques
- Initial signal partitioning can be a more general *overlap-add decomposition* (*e.g.*, Hann window with 50% overlap)
  - Needed for *nonlinear STFT processing* (not needed for simple FFT convolution)

Overlap-Add Signal Decomposition

Consider breaking the input signal x, into frames using a finite, zero-centered, length M (odd) window. Let x_m denote the m^{th} frame.

$$x_m(n) \overset{\Delta}{=} x(n)w(n-mR) \qquad n \in (-\infty, +\infty)$$

or

$$x_m \overset{\Delta}{=} x \cdot \text{SHIFT}_{mR}(w)$$

where

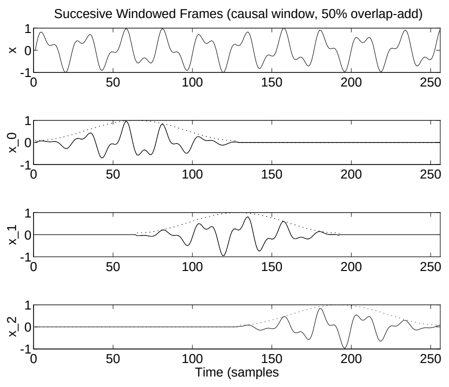
$R \overset{\Delta}{=}$ frame step (hop size) $\qquad m \overset{\Delta}{=}$ frame index

The *hop size* is the number of samples between adjacent frames. Specifically, it is the number of samples by which we advance each succesive window.

For fast convolution only, choose

- $w = w_R =$ rectangular window
- $R = M$ (hop size = window length)

Example Overlap-Add Decomposition



We desire the sum of overlapping frames x_m to give back the original input signal x:

$$x(n) = \sum_{m=-\infty}^{\infty} x_m(n) = \sum_{m=-\infty}^{\infty} x(n)w(n-mR) = x(n)\sum_{m=-\infty}^{\infty} w(n-mR)$$

Hence, $x = \sum_m x_m$ if and only if

$$\boxed{\sum_m w(n-mR) = 1} \qquad \text{(COLA condition)}$$

Constant Overlap-Add (COLA) Constraint

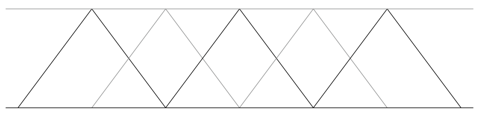
We found that $x = \sum_m x_m$ if and only if

$$\boxed{\sum_m w(n-mR) = 1} \qquad \text{(COLA condition)}$$

- This is the *constant-overlap-add* (COLA) constraint for the FFT analysis window w
- Also called the *partition of unity property*

COLA Windows

50% OLA for the Bartlett (triangular) window:

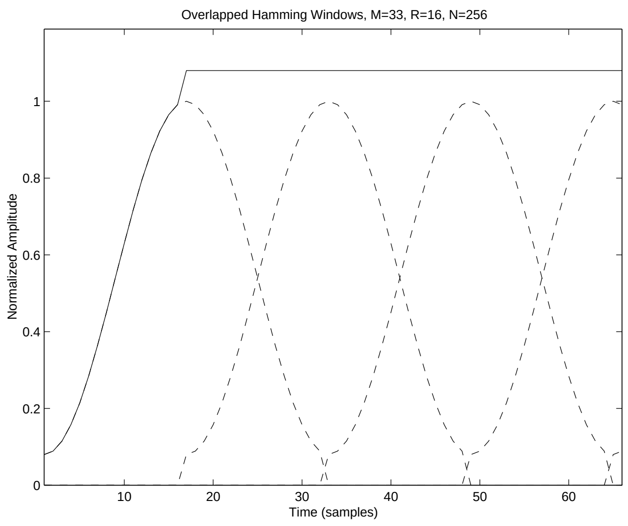


- All windows which obey the constant-overlap-add constraint will yield *perfect reconstruction* of the original signal from the data frames by *overlap-add* (OLA)
- There is no constraint on window type, only that the window overlap-adds to a constant for the hop size used
- If simple FIR filtering is being implemented, and we don't need a high-quality intermediate STFT, it is most efficient to use the *rectangular window* with hop size $R = M$, and to set $M + L - 1 = N$, where L is the length of the filter h
- The optimum window length M for a given filter length L is an interesting exercise to work out

Modified-Hamming Overlap-Add Example

Matlab code:

```
M = 33;          % window length
w = hamming(M);  % window
R = (M-1)/2;     % maximum hop size
w(M) = 0;        % 'periodic Hamming' (COLA)
```



COLA Periodic Hamming Window

Oversampled Perfect Reconstruction Filter Banks

Emphasis:

We now know how to make a fast,
Perfect-Reconstruction (PR) filter bank out of

- Hopping FFT
- Sufficient zero-padding
- COLA(R) window

However, we are not normally using *critically sampled* time and frequency axes (needed for compression)

COLA Window and Hop-Size Examples

- Recall COLA Examples[1] for the STFT
- Bartlett window at 50% overlap ($R \approx M/2$) (first figure above)
- Hamming window at 50% overlap ($R \approx M/2$) (second figure above)
- We will use the Poisson Summation Formula (derived below) to easily determine all COLA hop sizes for a given window

[1]https://ccrma.stanford.edu/~jos/TimeFreqDisplay/COLA_Examples.html

Periodic Windows

Be careful to define your own windows for exact overlap-add, or else at least check the definitions in use to make sure they work. E.g., in Matlab:

- `hamming(M)` $\stackrel{\Delta}{=}$
  `.54 - .46*cos(2*pi*(0:M-1)'/(M-1));`
  gives constant overlap-add for $R = (M-1)/2$, $(M-1)/4$, etc., when endpoints are divided by 2 or one endpoint is zeroed
- `hanning(M)` $\stackrel{\Delta}{=}$
  `.5*(1 - cos(2*pi*(1:M)'/(M+1)));`
  does *not* give constant overlap-add for $R = (M-1)/2$, but does for $R = (M+1)/2$
- `blackman(M)` $\stackrel{\Delta}{=}$
  `(.42 - .5*cos(2*pi*(0:M-1)/(M-1)) +`
  `.08*cos(4*pi*(0:M-1)/(M-1)))';`
  gives constant overlap-add for $R = (M-1)/3$ when M is odd and R is an integer, and $R = M/3$ when M is even and R is integer

Example 3: Overlap-Add Convolution

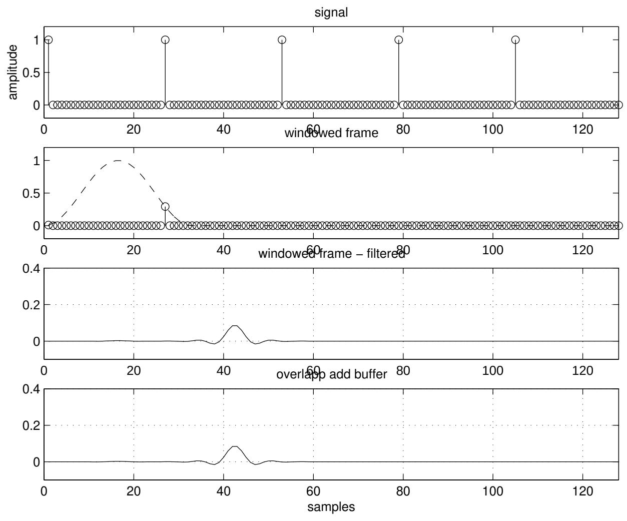
- Impulse-train signal, 4000 Hz sampling-rate
- Length 32 lowpass filter, 600 Hz cut-off
- Length $M = 32$ Hanning overlap-add window (causal)
- Hop size $= R = M/2$ (50% overlap)

Matlab code:

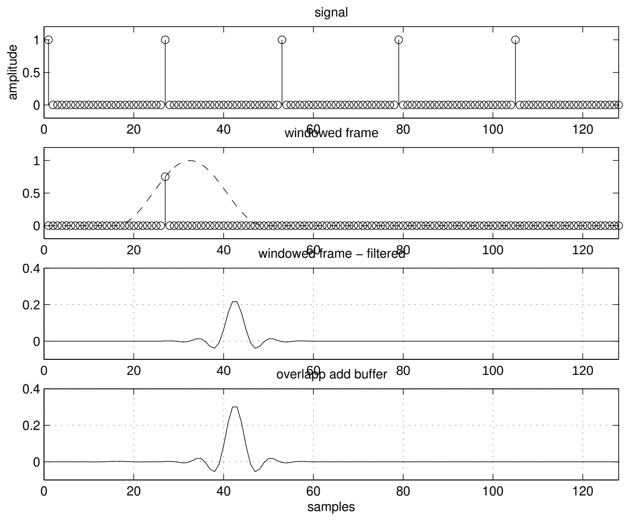
```
Fs = 4000;              % sampling rate in Hz
Nsig = 128;             % signal length in samples
L = 32;                 % filter length in taps
fc = 600;               % cutoff frequency in Hz
M = 32;                             % window length
Nfft = 2^(ceil(log2( M + L - 1))); % FFT Length
R = M/2;                            % Hop Size
Nframes = floor( Nsig/R );          % No. of frames

% Generate a test signal containing five impulses
sig = zeros(1,Nsig); period = round(Nsig/5);
sig(1:period:Nsig) = ones(size(1:period:Nsig));

figure(1);clf
subplot(211); stem(sig); axis( [0 length(sig) -0.2 1.2]);
subplot(212); stem(h); grid; axis( [0 length(h) -0.2 0.5]);

% *** design a lowpass filter using the window method
epsilon = .0001; % avoids 0 / 0
nfilt = (-(L-1)/2:(L-1)/2) + epsilon;
```

```
hideal = sin( 2*pi*fc*nfilt/Fs ) ./ (pi*nfilt); % sinc fn
h = hamming( L )' .* hideal; % windowed sinc fn

hzp = [h zeros(1,Nfft-L)];  % Zero-pad h to FFT size
H = fft(hzp);               % Filter frequency response
y = zeros(1,Nsig + Nfft);   % Allocate the output+'ringing' vector

% *** Overlap add loop, with plots
figure(2);
subplot(411); stem(sig); axis( [0 length(sig) -0.2 1.2]);
title('signal'); ylabel('amplitude');
window = hanning(M)';       % Signal window (could also be rect)
for m = 0:(Nframes-1)
    index = m*R+1:min(m*R+M,Nsig);      % index for the mth frame
    xm = sig(index) .* window(index-m*R); % windowed mth frame
    xmzp = [ xm zeros(1,Nfft-length(xm))];      % zero pad the signal
    Xm = fft( xmzp );
    Xfilt = Xm .* H;                    % freq domain multiplication
    xmfilt = real(ifft(Xfilt));         % inverse transform
    outindex = m*R+1:(m*R+Nfft);        %
    y(outindex) = y(outindex) + xmfilt; % overlap add

    xmplot = [ zeros(1,m*R) xm zeros(1,Nsig-(m*R+M)) ];
    winplot = [ zeros(1,m*R) window zeros(1,Nsig-(m*R+M)) ];
    xmfiltplot = [ zeros(1,m*R) xmfilt zeros(1,Nsig-(m*R+Nfft)) ];
    subplot(412);stem(xmplot); axis( [0 length(sig) -0.2 1.2]);
    hold on; plot(winplot,'--'); hold off
    title('windowed frame');
    subplot(413);plot(xmfiltplot); grid; axis( [0 length(sig) -0.1 0.4]);
    title('windowed frame - filtered');
    subplot(414);plot(y); grid; axis( [0 length(sig) -0.1 0.4]);
    title('overlapp add buffer'); xlabel('samples');
    cmd = sprintf('print -deps ola%d',m); disp(cmd); eval(cmd);
end
```

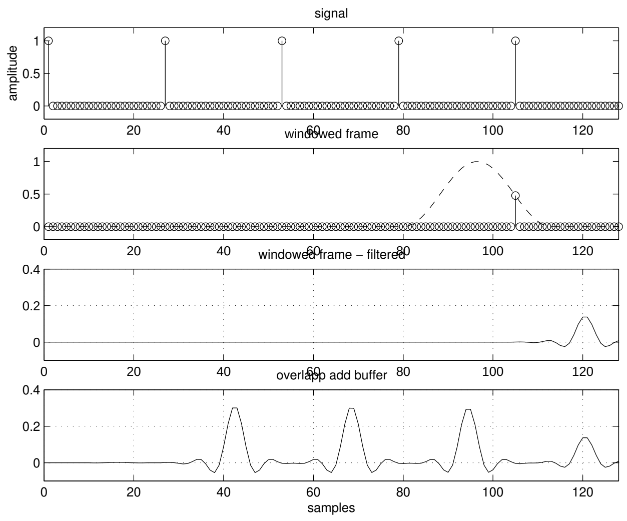
Frame 1, OLA Example 3



Frame 2, OLA Example 3

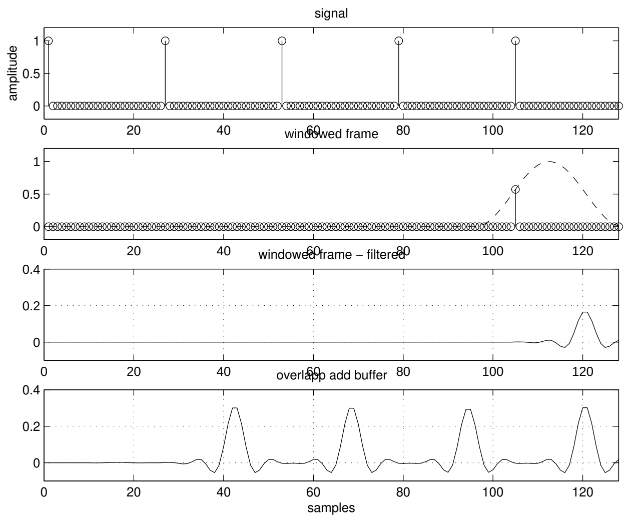


Penultimate Frame, OLA Example 3



29

Final Frame, OLA Example 3



30

OLA Summary

- OLA provides an efficient method for implementing LTI systems having impulse responses over ≈ 100 samples
- FFT results in minimized computations

Specifically, we ended up with:

$$\boxed{y = \sum_{m=-\infty}^{\infty} \textsc{Shift}_{mR}\left(\text{DFT}_N^{-1}\left\{H \cdot \text{DFT}_N\left[\textsc{Shift}_{-mR}(x)\cdot w\right]\right\}\right)}$$

(Shifts acyclic here)

31

OLA Summary, Cont'd

Stated as a procedure, we have

(1) Extract the mth frame of data at time mR.

(2) Shift it to the base time interval $[0, M-1]$ (or $[-(M-1)/2, (M-1)/2]$).

(3) Apply the analysis window w (causal or zero centered, as preferred).

(4) Zero-pad the windowed data out to the FFT size (a power of 2).

(5) Take the N-point FFT.

(6) Apply the filter H as a windowing operation in the frequency domain.

(7) Take the N-point inverse FFT.

(8) Shift the origin of the N-point result out to sample mR where it belongs.

(9) Sum into the output buffer containing the results from prior frames (OLA step).

32

OLA Summary, Cont'd

There were several conditions for exact FIR filtering:

- To avoid time domain aliasing: $N \geq M + L - 1$
  - M = window length
  - N = DFT length
  - L = FIR filter length
  - Equivalent to a minimum sampling-rate requirement in the frequency domain
- $\sum_m w(n - mR) = 1$ (COLA constraint)

Quick Summary of Zero Padding

Why zero pad?

- Spectral Interpolation
  - Clearer spectral magnitude/phase plots
  - Sinusoidal peak tracking (e.g. to help quadratic interpolation)
- To extend to the next highest power of 2 (FFT)
- To make room for "filter ringing" in overlap-add convolution using the FFT

Overlap-Save Method

The *OverLap-Save* (OLS) method, unlike OLA, uses *no zero padding* to prevent time aliasing. Instead, it

1. discards output samples corrupted by time aliasing each frame, and
2. overlaps the input frames by the same amount.

- In general, if the input frame size is $M = N$ and the FIR filter length is $L < N$, then after convolution, $L - 1$ samples of the output are invalid due to time aliasing
- For causal filters of length L:
  - The invalid samples are the first $L - 1$ samples of each length N inverse FFT, because these samples are computed using time-aliased samples from the end of the length N FFT input frame
  - Therefore, the input signal should have at least $L - 1$ leading zeros
  - The hop size is set to $R = N - L + 1$ so that the last $L - 1$ samples of frame 1 become the first segment of frame 2. These samples have already

been output from frame 1 and can now be overwritten by time aliasing by the processing of frame 2.
  - The length N blocks overlap by $L - 1$ samples. $L - 1$ samples from the previous block are "saved" rather than reread from disk—hence the name "OverLap Save (OLS)".
- For anticausal filters:
  - The invalid samples are at the *end* of the frame
  - The input signal needs no leading zeros
  - The hop size is again $R = N - L + 1$
  - Samples 0 through $R - 1$ are written out, ignoring the last $L - 1$ samples corrupted by time aliasing
- For general non-causal FIR filters, there are bad samples at the beginning and end of each inverse FFT.
- The overlap-save method is generally somewhat more efficient than the overlap-add method because it does not use zero padding, and there is no overap-add on output
- **Exercise:** Compare the computational complexities of overlap-add and overlap-save for the case

$N = M + L - 1$, where M is the frame length, L is the filter length, and N is the FFT size. Show that zero-phase filters yield the most efficient overlap-save scheme for a given filter length L.

Time Varying Modifications

In FFT convolution using the STFT, the filter can be changed each frame ($h \to h_m$):

- $X_m(\omega_k)$ = sampled DTFT (FFT) of m^{th} input frame
- $H_m(\omega_k)$ = time varying spectral modification
- $Y_m(\omega_k) = X_m(\omega_k)H_m(\omega_k) = m$th output frame
- $\omega_k = 2\pi k/N = k$th spectral sample
- N = FFT length
- M = window w length: $x_m(n) = x(n)w(n-m)$
- L = max length of FIR filter h_m applied to each frame
- $N >= M + L - 1$ to avoid time aliasing in y_m

Using H_m in our OLA formulation with a hop size $R = 1$ results in:

$$\begin{aligned}
y(n) &= \sum_{m=-\infty}^{\infty} y_m(n) \\
&= \sum_{m=-\infty}^{\infty} \frac{1}{N}\sum_{k=0}^{N-1} X_m(\omega_k)H_m(\omega_k)e^{j\omega_k n} \\
&= \sum_{m=-\infty}^{\infty} \frac{1}{N}\sum_{k=0}^{N-1} \left[\sum_{l=-\infty}^{\infty} x(l)w(l-m)e^{-j\omega_k l}\right] H_m(\omega_k)e^{j\omega_k n} \\
&= \sum_{l=-\infty}^{\infty} x(l) \sum_{m=-\infty}^{\infty} w(l-m)\frac{1}{N}\sum_{k=0}^{N-1} H_m(\omega_k)e^{j\omega_k(n-l)} \\
&= \sum_{l=-\infty}^{\infty} x(l) \sum_{m=-\infty}^{\infty} w(l-m)h_m(n-l)
\end{aligned}$$

Letting $r \overset{\Delta}{=} n - l \Rightarrow l = n - r$ results in:

$$y(n) = \sum_{r=-\infty}^{\infty} x(n-r) \sum_{m=-\infty}^{\infty} h_m(r)w(n-r-m)$$

Let's examine $\displaystyle\sum_{m=-\infty}^{\infty} h_m(r)w(n-r-m)$:

- $h_m(r)$ describes the time variation of the r^{th} ***tap*** of the time-varying FIR filter
- $\sum_{m=-\infty}^{\infty} h_m(r)w[(n-r)-m] = [h_{(\cdot)}(r) * w](n-r)$ is a filtered version of the r^{th} tap $h_m(r)$. It is lowpass-filtered by w and delayed by r samples.
- Denote the rth time-varying, lowpass-filtered, delayed-by-r filter tap by $\hat{h}^w_{n-r}(r)$. This can be interpreted as the weighting in the output at time r of an impulse entering the time-varying filter at time $n - r$.

Using this, we get

$$\begin{aligned}
y(n) &= \sum_{r=-\infty}^{\infty} x(n-r)\hat{h}^w_{n-r}(r) \qquad (= x * \hat{h}^w \text{ if LTI}) \\
&= x(n)\hat{h}^w_n(0) \\
&\quad + x(n-1)\hat{h}^w_{n-1}(1) + x(n-2)\hat{h}^w_{n-2}(2) + \cdots \\
&\quad + x(n+1)\hat{h}^w_{n+1}(-1) + x(n+2)\hat{h}^w_{n+2}(-2) + \cdots
\end{aligned}$$

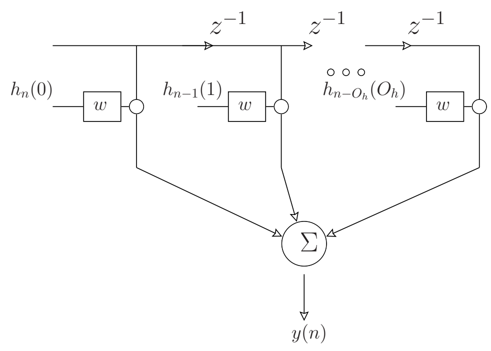
This is a superposition sum for an arbitrary linear, time-varying filter $\hat{h}^w_{n-r}(r) = [h_{(\cdot)}(r) * w](n-r)$.

Block Diagram Interpretation

Assuming $\hat{h}^w$ is causal gives

$$\begin{aligned} y(n) &= \sum_{r=0}^{\infty} x(n-r)\hat{h}^w_{n-r}(r) \\ &= x(n)\hat{h}^w_n(0) + x(n-1)\hat{h}^w_{n-1}(1) + x(n-2)\hat{h}^w_{n-2}(2) + \cdots \end{aligned}$$

This is depicted in the following diagram:



- $h_m(l)$ = FIR filter tap l at time m
- Each tap is *lowpass filtered* by the FFT window w
- The window thus enforces *bandlimiting* of the filter taps (see the text for further details)

Weighted Overlap Add

In the *weighted overlap add* (WOLA) method, we apply a second window after the inverse DFT. Such a window can be called a “synthesis window”, “post-window”, or simply “output window,” as opposed to the “analysis window”, “pre-window”, or “input window” we have been using up to now, prior to the FFT.

Synthesis windows are important in compression applications to minimize “blocking effects,” since spectral coding error is thereby “faded out” at the frame boundaries, preventing audible discontinuities.

Weighted OverLap Add (WOLA) Procedure

The sequence of operations in a WOLA processor can be expressed as follows:

1. Extract the mth windowed frame of data $x_m(n) = x(n)w(n-mR)$, $n = m, \ldots, m+N-1$ (assuming a length $M \le N$ causal window w and hop size R).
2. Take an FFT of the mth frame translated to time zero, $\tilde{x}_m(n) = x_m(n+mR)$, to produce the mth spectral frame $\tilde{X}_m(\omega_k)$, $k = 0, \ldots, N-1$.
3. Process $\tilde{X}_m(\omega_k)$ as desired to produce $\tilde{Y}_m(\omega_k)$.
4. Inverse FFT $\tilde{Y}_m$ to produce $\tilde{y}_m(n)$, $n = 0, \ldots, N-1$.
5. Apply a *synthesis window* $f(n)$ to $\tilde{y}_m(n)$ to yield a *weighted* output frame $\tilde{y}^f_m(n) = \tilde{y}_m(n)f(n)$, $n = 0, \ldots, N-1$.
6. Translate the mth output frame to time mR as $y^f_m(n) = \tilde{y}^f_m(n-mR)$ and add to the accumulated output signal $y(n)$.

- The overlap-add method is obtained from the above procedure by deleting step 5.

- To obtain perfect reconstruction in the absence of spectral modifications, we require

$$\begin{aligned} x(n) &= \sum_{m=-\infty}^{\infty} x(n)w(n-mR)f(n-mR) \\ &= x(n) \sum_{m=-\infty}^{\infty} w(n-mR)f(n-mR) \end{aligned}$$

which is true if and only if

$$\boxed{\sum_m w(n-mR)f(n-mR) = 1}$$

When to Use Weighted Overlap-Add

- Synthesis windows are not used in simple FFT convolution processors using the OLA method, since the input frames are *supposed* to be expanded by the convolution, and the synthesis window would "pinch off" the "filter ringing", yielding the wrong results.
- Synthesis windows *can* be used in conjunction with spectral modifications made by means of the "filter bank summation" (FBS) method, which is the subject of the next lecture.
- Synthesis windows are appropriate for erstwhile "instantaneous" spectral modifications, such as
  - spectral quantization (compression),
  - frequency scaling (time-scale modification), or
  - "memoryless" spectral nonlinearities.

In other words, whenever samples outside of the original signal time frame are considered processing artifacts, use of a synthesis window will eliminate them.

Choice of WOLA Window

The synthesis window in weighted overlap-add is typically chosen to be the same as the input window, in which case the COLA constraint becomes

$$\boxed{\sum_m w^2(n-mR) = \text{constant}}$$

We can say that R-shifts of the window w in the time domain are *power complementary* (power-partition of unity), whereas for OLA they were *amplitude complementary*.

WOLA Window Construction

A trivial way to construct useful windows for WOLA is to take the *square root* of any good OLA window. This works for all non-negative OLA windows (which includes essentially all typical windows).

- For example, the "root-Hann window" can be defined for odd M by

$$\begin{aligned} w(n) &= w_R(n)\sqrt{\frac{1}{2}+\frac{1}{2}\cos(2\pi n/M)} \\ &= w_R(n)\cos(\pi n/M),\ n=-\frac{M-1}{2},\dots,\frac{M-1}{2} \end{aligned}$$

- Notice that the root-Hann window is essentially the same thing as the "MLT Sine Window" described in the lecture on FFT windows.
- Similarly, we can define the "root-Hamming", "root-Blackman", and other windows for perfect reconstruction in the weighted overlap-add context.

WOLA Applications

Nonlinear "instantaneous" FFT processors such as

- perceptual audio coders,
- time-scale modification, or
- pitch-shift (frequency-scaling)

are normally based on the Weighted Overlap-Add (WOLA) method for short-time Fourier analysis, modification, and resynthesis because

- the synthesis window (applied after the inverse FFT and prior to overlap-add reconstruction) helps to suppress *artifacts* caused by nonlinear spectral modifications, and
- filtering effects are not desired, so no provision for "ringing" in the time domain is needed.

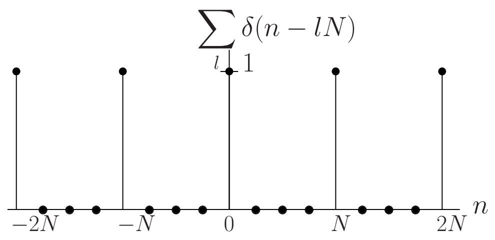
Poisson Summation Formula Derivation

First consider the summation of N complex exponentials:

$$x(n) \triangleq \frac{1}{N}\sum_{k=0}^{N-1} e^{j\omega_k n} = \begin{cases} 1 & n = 0 \pmod{N} \\ 0 & \text{elsewhere} \end{cases}$$

$$= \text{IDFT}_{N,n}(1\cdots 1) = \delta(n) + \delta(n-N) + \delta(n+N) + \cdots$$

where $\omega_k \triangleq 2\pi k/N$.

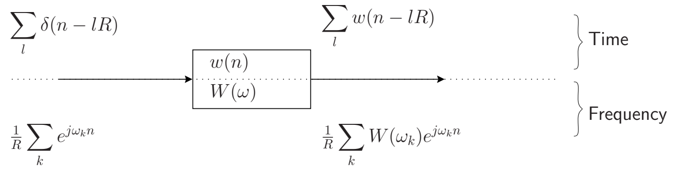


Setting $N = R$ (the FFT hop size) gives

$$\boxed{\sum_m \delta(n-mR) = \frac{1}{R}\sum_{k=0}^{R-1} e^{j\omega_k n}}$$

where $\omega_k \triangleq 2\pi k/R$ (harmonics of the *frame rate*).

Let us now consider these equivalent signals as inputs to an LTI system, with an impulse response given by $w(n)$, and frequency response equal to $W(\omega)$.



$\sum_l \delta(n-lR)$
$\sum_l w(n-lR)$
Time
$w(n)$
$W(\omega)$
$\frac{1}{R}\sum_k e^{j\omega_k n}$
$\frac{1}{R}\sum_k W(\omega_k)e^{j\omega_k n}$
Frequency

Looking across the top of the above figure, for the case of input signal $x(n) = \sum_m \delta(n-mR)$ we have:

$$y(n) = \sum_m w(n-mR)$$

and looking across the bottom of the above figure, for the case of input signal $\frac{1}{R}\sum_{k=0}^{R-1} e^{j\omega_k n}$, we have:

$$y(n) = \frac{1}{R}\sum_{k=0}^{R-1} W(\omega_k)e^{j\omega_k n}$$

Since the inputs were equal, the corresponding outputs must be equal too. This derives the Poisson Summation Formula:

Poisson Summation Formula

$$\boxed{\underbrace{\sum_m w(n-mR)}_{\text{ALIAS}_R(w)} = \underbrace{\frac{1}{R}\sum_{k=0}^{R-1} W(\omega_k)e^{j\omega_k n}}_{\text{DFT}_R^{-1}\left[\text{SAMPLE}_{\frac{2\pi}{R}}(W)\right]}} \qquad \omega_k \triangleq \frac{2\pi k}{R}$$

- Dual of the sampling theorem
- COLA $\equiv W(\omega_k) = 0, \quad |k| = 1, 2, \ldots, R-1$
- In other words, *constant overlap-add of w at hop-size R happens if and only if the window transform W is zero at the frame rate $2\pi/R$ and all its harmonics.*
- **Notation:**

  $$\boxed{w \in \text{COLA}(R) \quad \Leftrightarrow \quad W \in \text{NYQUIST}(2\pi/R)}$$

  The "Nyquist(Ω_R)" property for a function W simply means that W is zero at all nonzero multiples of Ω_R (all harmonics of the frame rate here).

When the COLA condition is met, we have, by the PSF,

$$\boxed{\sum_m w(n-mR) = \frac{1}{R}W(0)}$$

Strong COLA

An *overly strong* (but sufficient) condition is to bandlimit $W(\omega)$ consistent with downsampling by R:

$$W(\omega) = 0, \quad |\omega| \geq \pi/R \qquad \text{(sufficient for COLA)}$$

Weak COLA

The *necessary and sufficient* condition for $w \in \text{COLA}(R)$ is again *zeros at all frame-rate harmonics:*

$$W(\omega_k) = 0, \quad \omega_k = k\frac{2\pi}{R}, \; k = \pm 1, \pm 2, \ldots \quad \text{(nec. \& suff.)}$$

PSF Dual and Graphical Equalizers

PSF:

$$w \in \textsc{Cola}(R) \quad \Leftrightarrow \quad W \in \textsc{Nyquist}(2\pi/R)$$

PSF Dual:

$$W \in \textsc{Cola}(2\pi/N) \quad \Leftrightarrow \quad w \in \textsc{Nyquist}(N)$$

Interpretation:

- $N =$ number of (complex) filters in a *filter bank*
- Passbands uniformly distributed around the unit circle
- Typical implementation: FFT filter bank

Let $W(e^{j\omega T}) =$ "dc channel" of filter bank. Then $W \in \textsc{Cola}(2\pi/N)$ means

$$S(\omega) = \sum_{k=0}^{N-1} W\left(e^{j(\omega-\omega_k)T}\right) = \text{constant}$$

where $\omega_k T \;\stackrel{\Delta}{=}\; k\Omega_N \;\stackrel{\Delta}{=}\; k \cdot 2\pi/N$.

In the time domain, $w \in \textsc{Nyquist}(N)$ means that w is zero at all nonzero integer multiples of N:

$$w(n) = 0, \quad n = \pm N, \pm 2N, \pm 3N, \ldots$$

Ideal Graphical Equalizers

PSF Dual:

$$W \in \textsc{Cola}(2\pi/N) \quad \Leftrightarrow \quad w \in \textsc{Nyquist}(N)$$

- N-channel equalizer filter banks can be made using bandpass filters having zero-crossings at multiples of N samples
- All such filter banks sum to a constant frequency response when channel gains are equal
- The PSF dual can be taken as the basis for the *Filter-Bank Summation* (FBS) interpretation of the short-time Fourier transform
- The FBS is precisely the Fourier dual of the OverLap-Add (OLA) interpretation