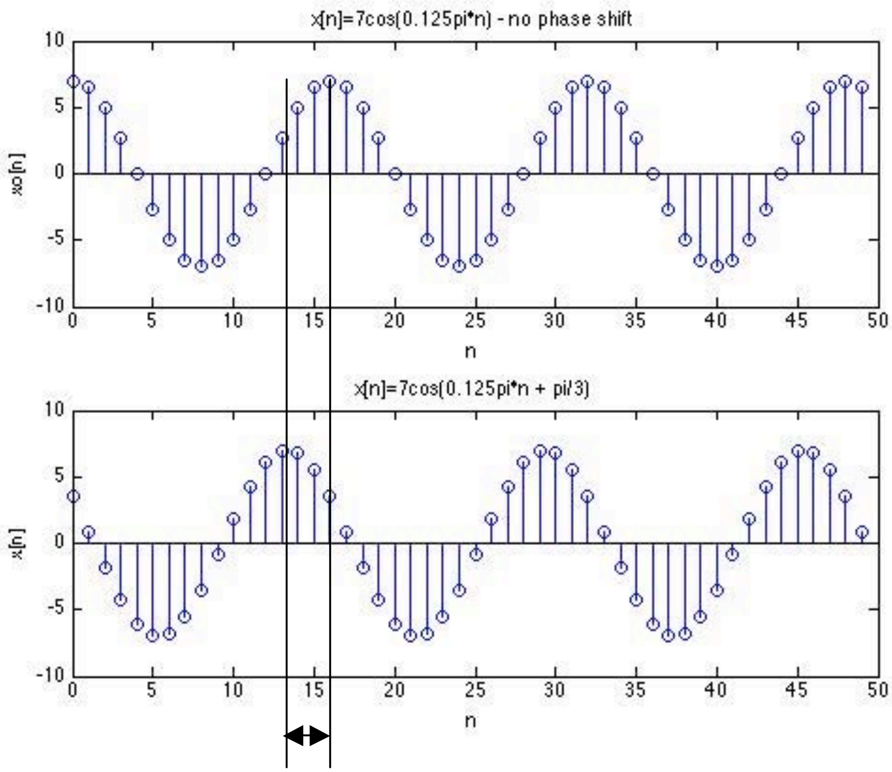
MAS.160 / MAS.510 / MAS.511 Signals, Systems and Information for Media Technology
Fall 2007

phase



$7\cos\left(\frac{\pi}{8}n\right)$

$7\cos\left(\frac{\pi}{8}n + \frac{\pi}{3}\right)$

$$\hat{\omega} = \frac{\pi}{8} = \frac{2\pi}{16} = 2\pi f \Rightarrow f = \frac{1}{16}$$

$$T = \frac{1}{f} = 16 \quad samples/cycle$$

phase is always relative

2.667 sample shift

$$2.667 \quad samples \cdot \frac{2\pi \quad rad/cycle}{16 \quad samples/cycle} = \frac{\pi}{3} rad$$

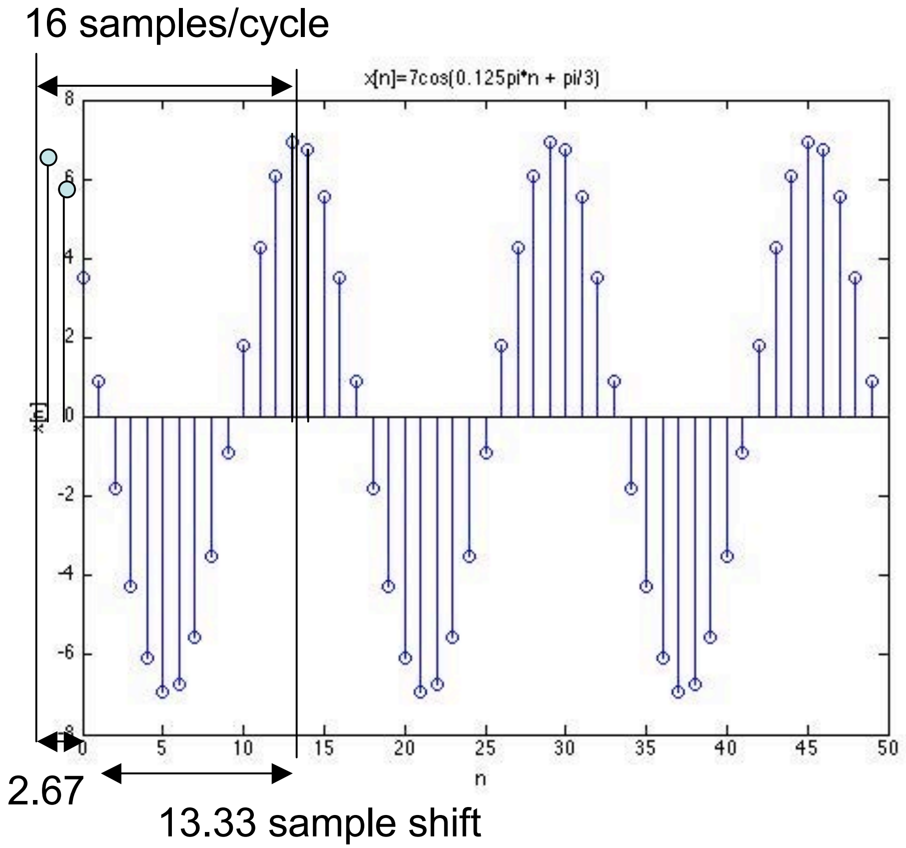
here, compare peaks of
shifted and unshifted cosines

peak occurs between samples, so interpolate

shifted peak occurs before unshifted peak,
so lead, so $+\phi$

phase

phase just from plot
(unshifted not plotted)



$7\cos\left(\frac{\pi}{8}n+\frac{\pi}{3}\right)$

$$(16-13.33)\quad samples\cdot\frac{2\pi \quad rad/cycle}{16 \quad samples/cycle}=\frac{\pi}{3}rad$$

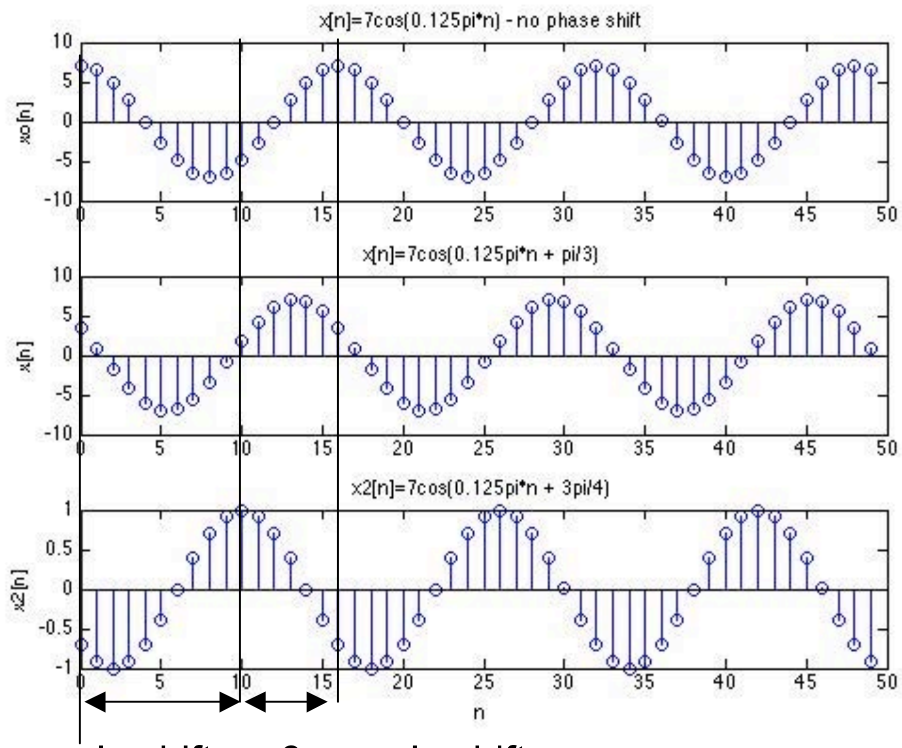
$$2.667 \quad samples\cdot\frac{2\pi \quad rad/cycle}{16 \quad samples/cycle}=\frac{\pi}{3}rad$$

here, look at “missing” sample shift to peak of cosine

shifted peak occurs before unshifted peak, so lead, so $+\phi$

relative phase

phase between two shifted cosines



$7\cos\left(\frac{\pi}{8}n\right)$

$7\cos\left(\frac{\pi}{8}n + \frac{\pi}{3}\right)$

$7\cos\left(\frac{\pi}{8}n + \frac{3\pi}{4}\right)$

10 sample shift 6 sample shift

estimate phase of x2

$$(16-10)\ \ samples \cdot \frac{2\pi\ \ rad/cycle}{16\ \ samples/cycle} = \frac{3\pi}{4} rad$$

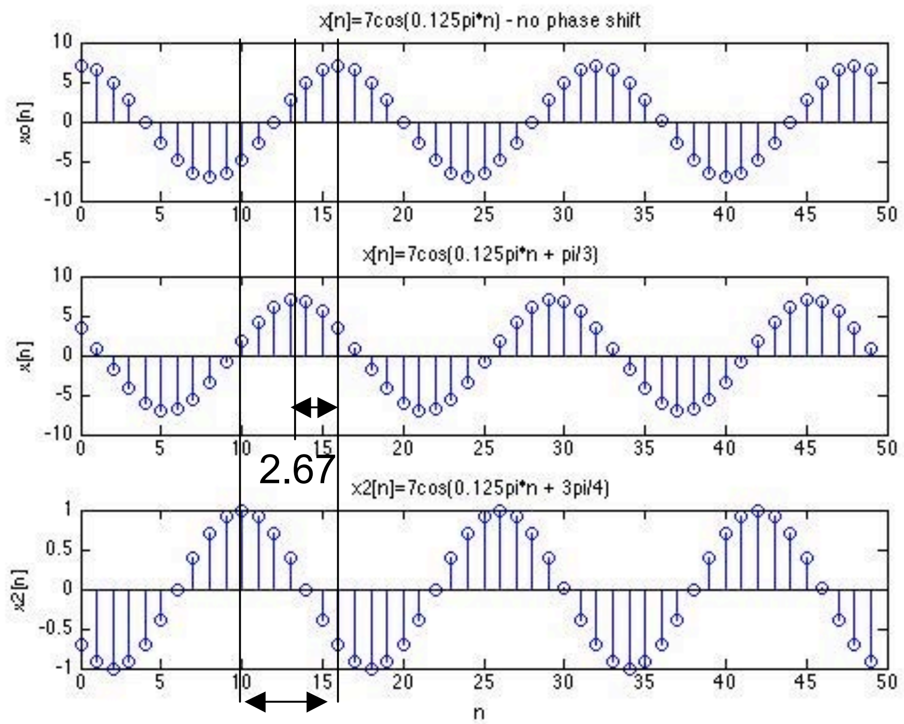
$$6\ \ samples \cdot \frac{2\pi\ \ rad/cycle}{16\ \ samples/cycle} = \frac{3\pi}{4} rad$$

here, look at “missing” sample shift to peak of cosine

shifted peak occurs before unshifted peak, so lead, so $+\phi$

relative phase

phase between two shifted cosines



6 sample shift

$7\cos\left(\frac{\pi}{8}n\right)$

$7\cos\left(\frac{\pi}{8}n+\frac{\pi}{3}\right)$

$7\cos\left(\frac{\pi}{8}n+\frac{3\pi}{4}\right)$

Subtract phases, ϕ_{x2}- ϕ_x

$$\frac{3\pi}{4}-\frac{\pi}{3}=\frac{5\pi}{12}rad$$

compare

$$(6-2.67)samples\cdot\frac{2\pi}{16}\frac{rad/cycle}{samples/cycle}=\frac{5\pi}{12}rad$$

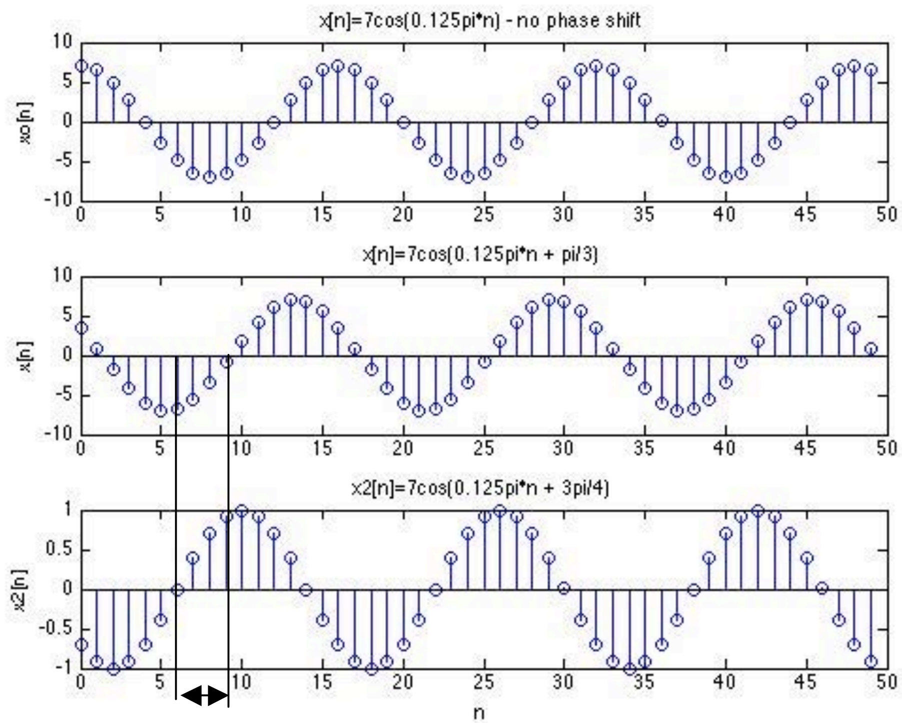
$$(3.33)samples\cdot\frac{2\pi}{16}\frac{rad/cycle}{samples/cycle}=\frac{5\pi}{12}rad$$

$$\left(\frac{10}{3}\right)samples\cdot\frac{2\pi}{16}\frac{rad/cycle}{samples/cycle}=\frac{5\pi}{12}rad$$

relative phase

phase between two shifted cosines

look at shift between zero-crossings



$7\cos\left(\frac{\pi}{8}n\right)$

$7\cos\left(\frac{\pi}{8}n + \frac{\pi}{3}\right)$

$7\cos\left(\frac{\pi}{8}n + \frac{3\pi}{4}\right)$

3.33 sample shift

zero-crossing occurs between samples, so interpolate.

easier to interpolate zero-crossing, than peak.

$$(3.33) samples \cdot \frac{2\pi}{16} \frac{rad/cycle}{samples/cycle} = \frac{5\pi}{12} rad$$

Systems

represent the system

solve system response to arbitrary input

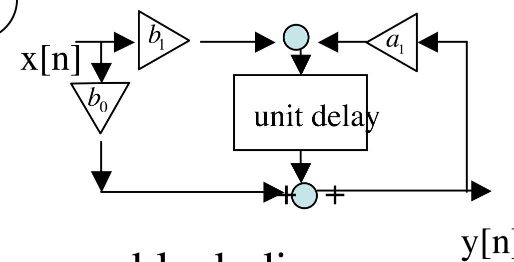
Equivalent ways to represent the system

1

$$y[n] = \sum_{l=1}^{N} a_l y[n-l] + \sum_{k=0}^{M} b_k x[n-k] \quad \Leftrightarrow$$

difference equation

2



block diagram

$x[n]=\delta[n]$ $\Updownarrow$

$\Updownarrow$ inspection

3

$$h[n] = y[n]\Big|_{x[n]=\delta[n]} \overset{Z}{\Leftrightarrow} H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 - \sum_{k=1}^{N} a_k z^{-k}} = \frac{\prod_{i=0}^{M}(z - z_{zi})}{\prod_{i=0}^{N}(z - z_{pi})}$$

impulse response sequence

4 system function polynomial

pole-zero locations 5

$z = e^{j\omega}$ $\Updownarrow$

6

$$\mathcal{H}(\omega) = H\left(e^{j\omega}\right) = H(z)\Big|_{z=e^{j\omega}}$$

frequency response

The region of convergence must Contain the unit circle for $\mathcal{H}(\omega)$ to converge and the system to be stable. (general)
(FIR filter always stable)

Equivalent ways to solve for response to arbitrary input

1) $$y[n] = \sum_{l=1}^{N} a_l y[n-l] + \sum_{k=0}^{M} b_k x[n-k]$$

iteration of difference equation

2) $$y[n] = h[n] * x[n]$$

convolve input with impulse response

$$h[n] = y[n]\Big|_{x[n]=\delta[n]}$$

impulse response

3) $$y[n]\Big|_{x[n]=e^{j\hat{\omega}n}} = \mathcal{H}(\hat{\omega})e^{j\hat{\omega}n}$$

use frequency response

$$\mathcal{H}(\omega) = H\left(e^{j\omega}\right) = H(z)\Big|_{z=e^{j\omega}}$$

frequency response

4) $$Y(z) = H(z) \cdot X(z)$$

$$\Downarrow IZT$$

$$y[n]$$

use z-transforms

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 - \sum_{k=1}^{N} a_k z^{-k}}$$

z-transform of diff.eqn.

Equivalent ways to solve for response to arbitrary input

1 $y[n] = \sum_{l=1}^{N} a_l y[n-l] + \sum_{k=0}^{M} b_k x[n-k]$

Initial rest conditions

iteration of difference equation

2 $y[n] = h[n] * x[n]$ LTI

convolve input with impulse response

FIR $h[n]=b_n$

IIR $h[n]$ solved iteratively (see 1)

a) convolution sum
- i) numerical
- ii) graphical

b) synthetic polynomial multiplication
- i) numerical
- ii) graphical

3 $y[n]\big|_{x[n]=e^{j\hat{\omega}n}} = \mathcal{H}(\hat{\omega})e^{j\hat{\omega}n}$

use frequency response

a) FIR $\longrightarrow$ $\mathcal{H}(\hat{\omega}) = \sum_{k=0}^{M} h[k]e^{j\hat{\omega}k} = \sum_{k=0}^{M} b_k e^{j\hat{\omega}k}$

b) FIR/IIR

$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 - \sum_{k=1}^{N} a_k z^{-k}}$ $\mathcal{H}(\omega) = H(e^{j\omega}) = H(z)\big|_{z=e^{j\omega}}$

$\mathcal{H}(\omega) = H(e^{j\omega})$ *exists?*

look at poles / roc.

roc must contain unit circle

4 $y[n] = h[n] * x[n]$

$\Downarrow ZT$

$Y(z) = H(z) \cdot X(z)$

$\Downarrow IZT$

$y[n]$

use z-transforms

Inverse z-transform

a) long division

b) lookup table

c) partial fraction
- i) match coeff
- ii) powers of z
- iii) powers of z^{-1}

		roc
right sided sequence (causal)		
$\frac{1}{1-az^{-1}} \Leftrightarrow a^n u[n]$		$\lvert z\rvert > \lvert a\rvert$
left sided sequence		
$\frac{1}{1-az^{-1}} \Leftrightarrow -a^n u[-n-1]$		$\lvert z\rvert < \lvert a\rvert$

Equivalent ways to solve for response to arbitrary input

1 $$y[n] = \sum_{l=1}^{N} a_l y[n-l] + \sum_{k=0}^{M} b_k x[n-k]$$

iteration of difference equation

one sample at a time
(possibly in sequence order)
Do you have eqn?

2 $$y[n] = h[n] * x[n]$$

convolve input with impulse response

one sample at a time
(possibly in sequence order)
Do you have impulse response?

3 $$y[n]\Big|_{x[n]=e^{j\hat{\omega}n}} = \mathcal{H}(\hat{\omega})e^{j\hat{\omega}n}$$

use frequency response

one frequency component
at a time.
Do you know freq content of x[n]?

4 $$Y(z) = H(z) \cdot X(z)$$

$$\Downarrow IZT$$

$$y[n]$$

use z-transforms

General
Can you do inverse z-transform?

Fourier Transforms

Compute spectrum of signals

Fourier Series	$X_k = \frac{2}{T_0} \int_0^{T_0} x(t) e^{-j2\pi kt/T_0} dt$	Periodic in (cont.) time Discrete freq
DTFT	$\mathcal{H}(\hat{\omega}) = \sum_{k=0}^{\infty} h[k] e^{j\hat{\omega}k}$	Discrete time Periodic in (cont.) freq
DFT	$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j(2\pi k/N)n}$	Discrete & periodic time Discrete & periodic freq

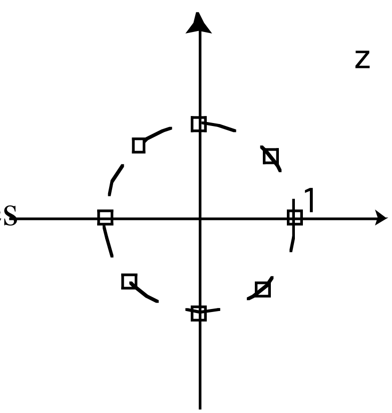
Discrete Fourier Transform (DFT)

Compute spectrum of discrete-time periodic signals

N samples in time domain $\underset{\text{IDFT}}{\overset{\text{DFT}}{\rightleftarrows}}$ N complex numbers in frequency domain

DFT $$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j(2\pi k/N)n}$$ analysis

IDFT $$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j(2\pi k/N)n}$$ synthesis



DFT: sample continuous H(ω) (DTFT) at N evenly spaced frequencies

Pad to get more samples / “bins”.
Window data. (DFT assumes periodicity).

FFT is an efficient algorithm to compute DFT

DFT Convolution

$$y[n] * x[n] \overset{\text{DTFT}}{\Leftrightarrow} Z(\hat{\omega}) = Y(\hat{\omega})X(\hat{\omega}) \overset{\text{IDTFT}}{\Leftrightarrow} z[n]$$

sample domain | frequency domain | sample domain

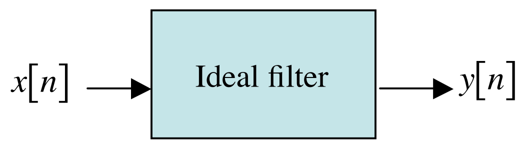
$Y[k]$ sampled version of $Y(\hat{\omega})$

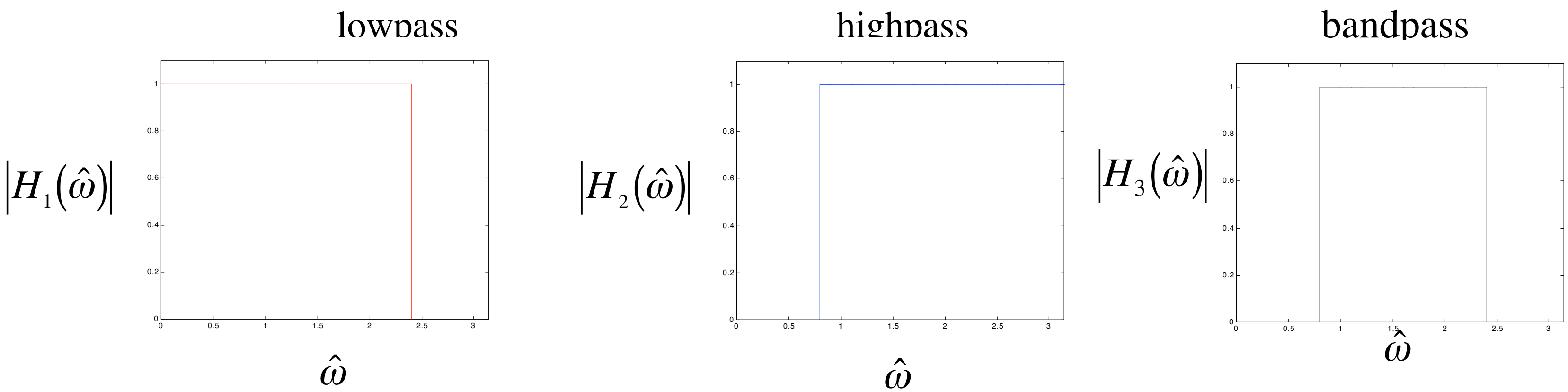
Use DFT to compute Y[k] and X[k]

$$\underbrace{y[n] \otimes x[n]}_{\text{circular convolution}} \overset{\text{DFT}}{\Leftrightarrow} Z[k] = Y[k]X[k] \overset{\text{IDFT}}{\Leftrightarrow} z[n]$$

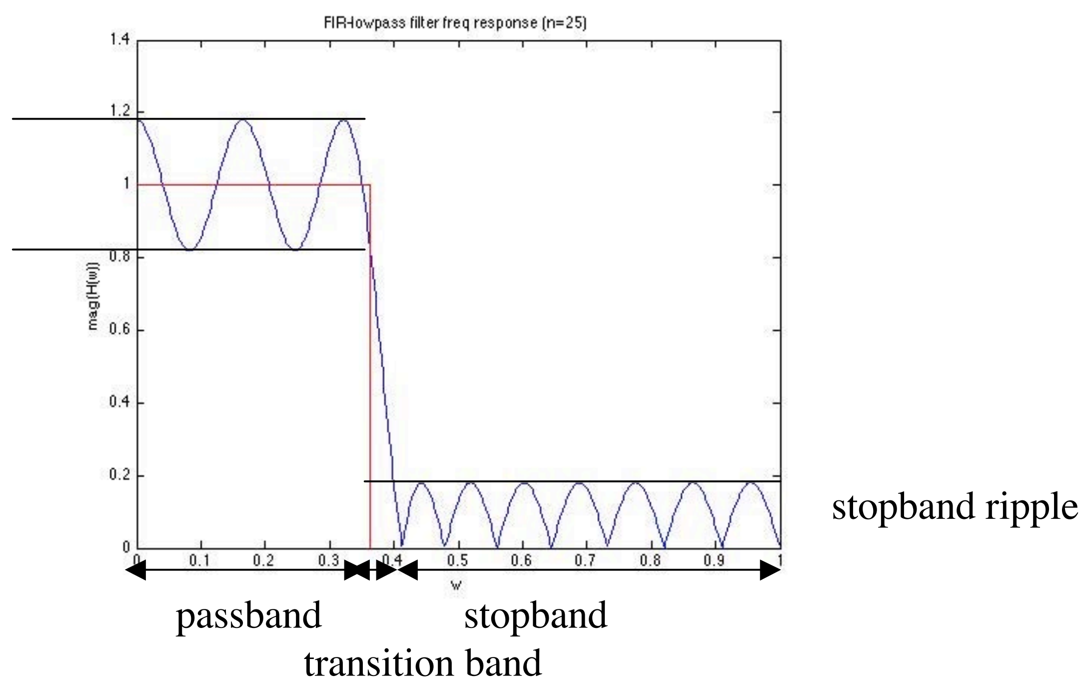
To avoid temporal aliasing:
if len(x)=N, len(y)=M, then pad so lengths are N+M-1

Filter Design





real filters



Filter Design

FIR vs. IIR Filters (Matlab Help)

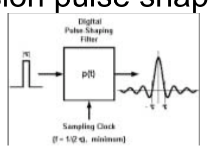
FIR filters advantages:
They can have exactly linear phase.
They are always stable.
The design methods are generally linear.
They can be realized efficiently in hardware.
The filter startup transients have finite duration.

FIR filter disadvantage
Much higher filter order than IIR filters to achieve a given level of performance.
Delay is greater than for an equal performance IIR filter.

Filter Design

FIR Filters Design Methods

Description	Matlab functions
Windowing Apply window to truncated inverse Fourier transform of desired "brick wall" filter	**fir1**, fir2, kaiserord
Multiband with Transition Bands Equiripple or least squares approach over sub-bands of the frequency range	firls, **firpm**, firpmord
Constrained Least Squares Minimize squared integral error over entire frequency range subject to maximum error constraints	fircls, fircls1
Arbitrary Response Arbitrary responses, including nonlinear phase and complex filters	cfirpm
Raised Cosine Lowpass response with smooth, sinusoidal transition (used for data transmission pulse shaping)	firrcos



FIR1 FIR filter design using the window method.

B = FIR1(N,Wn) designs an N'th order lowpass FIR digital filter and returns the filter coefficients in length N+1 vector B. The cut-off frequency Wn must be between 0 < Wn < 1.0, with 1.0 corresponding to half the sample rate.

B = FIR1(N,Wn,'high') highpass filter.
B = FIR1(N,Wn,'low') lowpass filter.

B = FIR1(N,Wn,'bandpass') if Wn = [W1 W2] with bandpass filter with passband W1 < W < W2.
B = FIR1(N,Wn,'stop') if Wn = [W1 W2] will design a bandstop filter.

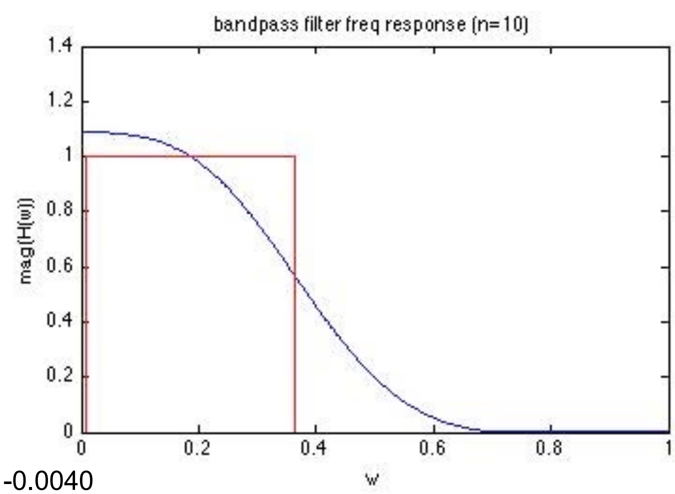
Ex. We have a sound recording (sampled at 44000Hz) and we want to isolate the speech range (200-8000Hz). (Note: we assume the sound recording was properly analog filtered BEFORE sampling to avoid aliasing. You can't remove the aliasing after you sample.)

Design a bandpass filter to isolate speech range.
We are sampling at 44kHz, so the maximum frequency in the recording is 22kHz (Nyquist).

$$f_{\max} = 22000Hz \Rightarrow w_{\max} = 1$$

$$f_1 = 200Hz \Rightarrow w_1 = \frac{200}{22000} = 0.0091$$

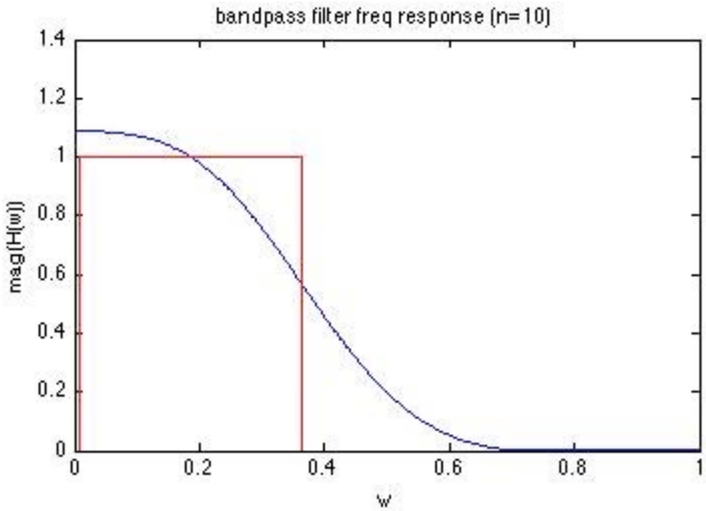
$$f_2 = 8000Hz \Rightarrow w_2 = \frac{8000}{22000} = \frac{4}{11} = 0.3636$$

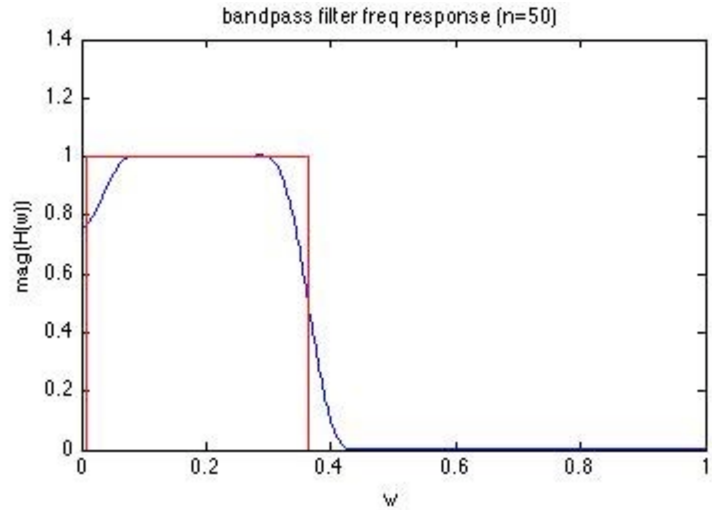


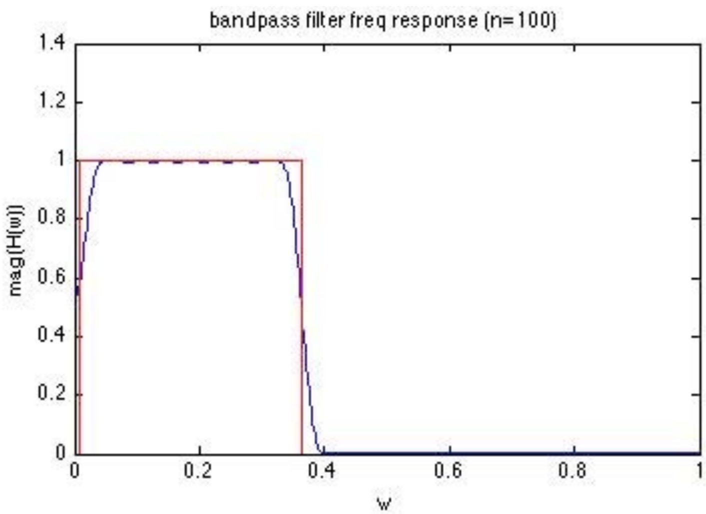
>>b=fir1(10,[0.0091 0.3636],'bandpass')

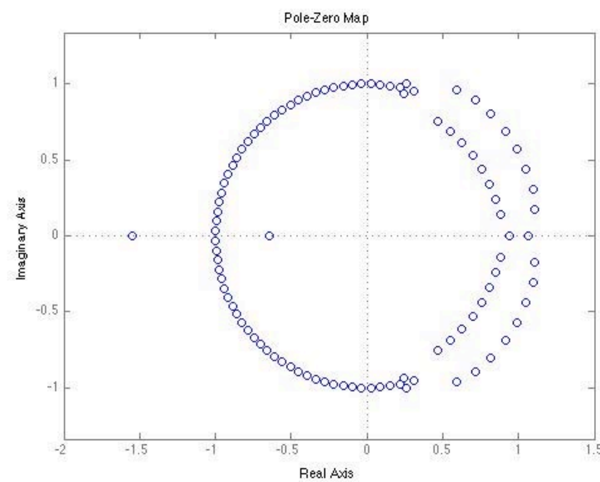
b =
-0.0040 -0.0169 -0.0177 0.0869 0.2930 0.4061 0.2930 0.0869 -0.0177 -0.0169 -0.0040

>>[h,w]=freqz(b,1);plot(w/pi,abs(h))







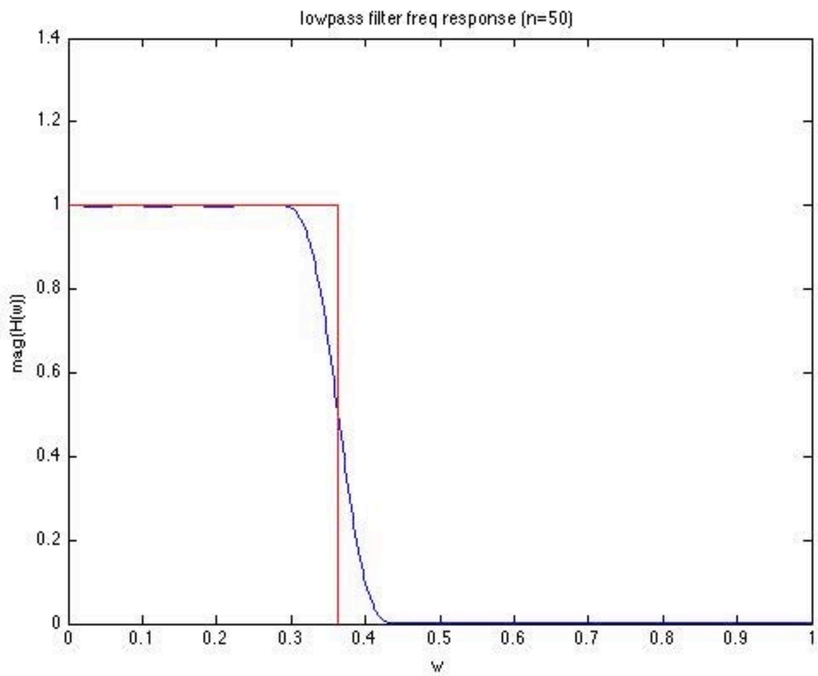


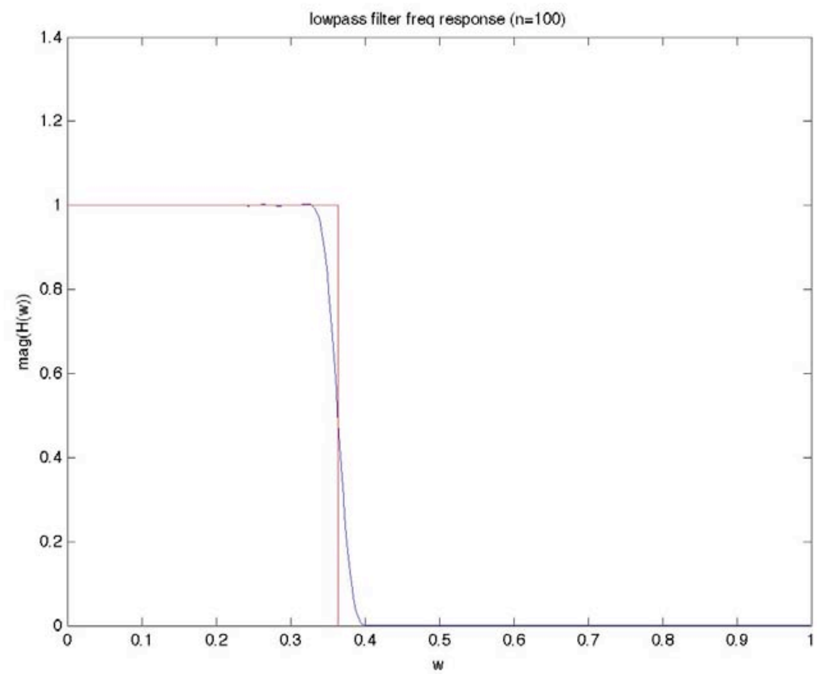
\>> pzmap(tf(b,1))

Realtime?
An optimized 20MHz MAXQ2000 uC can run a 100 tap FIR filter at close to 60kHz sampling rate.

http://www.maxim-ic.com/appnotes.cfm/an_pk/3483

LOW PASS






```
>>b=fir1(50,[0.0091 0.3636],'low')
>>[h,w]=freqz(b,1);plot(w/pi,abs(h))
```

```
>>b=fir1(100,[0.0091 0.3636],'low')
>>[h,w]=freqz(b,1);plot(w/pi,abs(h))
```

b = firpm(n,f,a)

order n FIR filter (n+1 coefficients)
linear-phase FIR filter
Uses Remez exchange algorithm
Maximum error between the desired and the actual frequency response is minimized.
Equiripple filters -- exhibit an equiripple behavior in their frequency responses

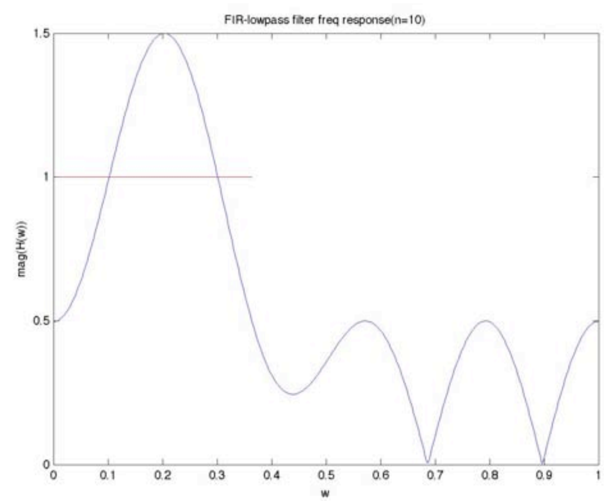
f and a specify the frequency-magnitude characteristics of the filter:
f is a vector of **pairs** of normalized frequency points, specified in the range between 0 and 1, frequencies must be in increasing order.

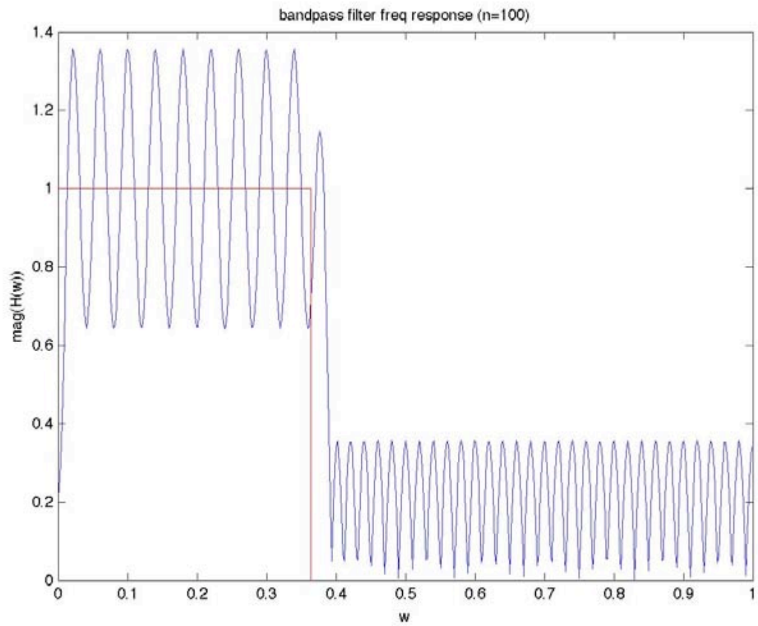
a is a vector containing the desired amplitudes at the points specified in f. (Between pairs, ‘don’t care’))

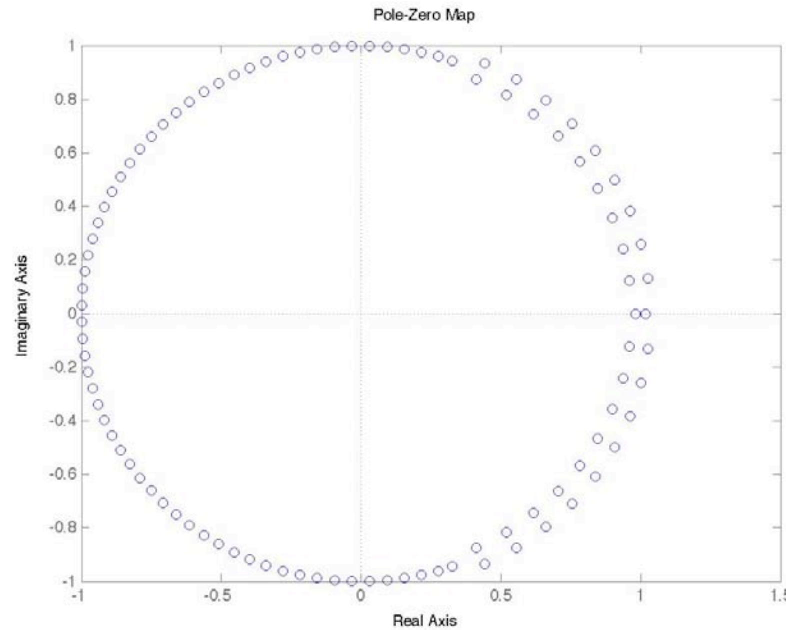
f and a must be the same length. The length must be an even number.

$$f_1 = 200Hz \Rightarrow w_1 = \frac{200}{22000} = 0.0091 \qquad f_2 = 8000Hz \Rightarrow w_2 = \frac{8000}{22000} = \frac{4}{11} = 0.3636 \qquad f_{\max} = 22000Hz \Rightarrow w_{\max} = 1$$

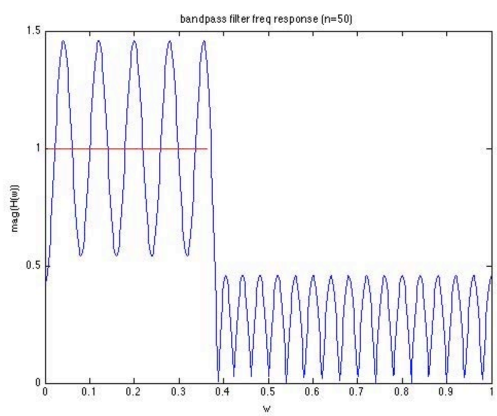
```
>>b=firpm(10,[0 0.005 0.0091  0.3636 0.39 1],[0 0 1 1 0 0]);
b = -0.2149   -0.0145   -0.0986    0.0353    0.3131    0.4559    0.3131
    0.0353   -0.0986 -0.0145   -0.2149

>>[h,w]=freqz(b,1);plot(w/pi,abs(h))
```



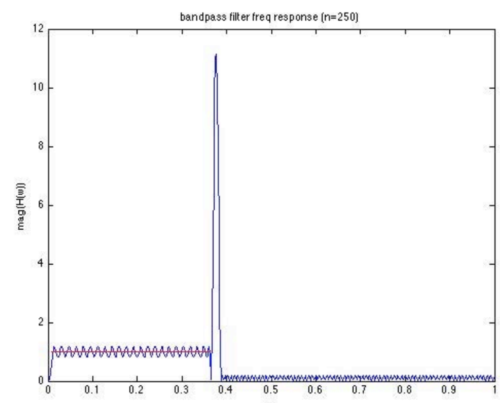




>>b=firpm(100,[0 0.005 0.0091 0.3636 0.39 1],[0 0 1 1 0 0]);
>>pzmap(tf(b,a))

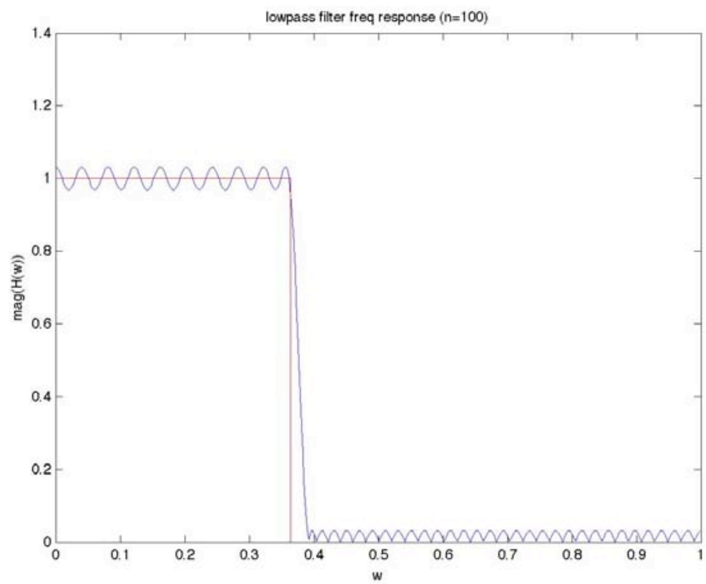


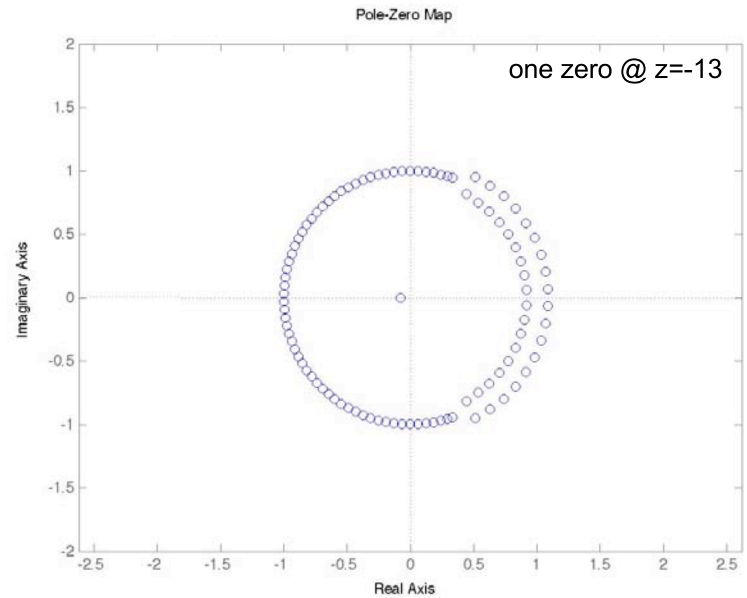
n=50



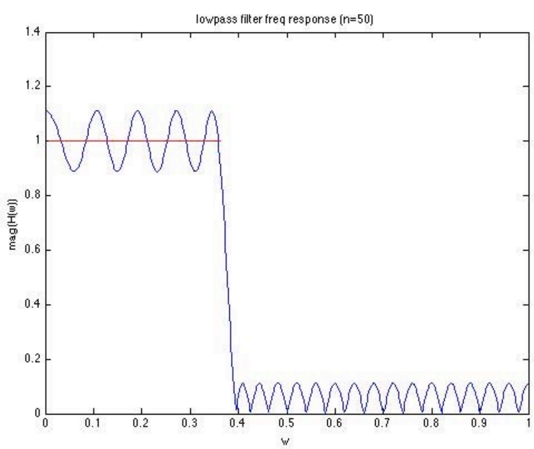
n=250

Lowpass

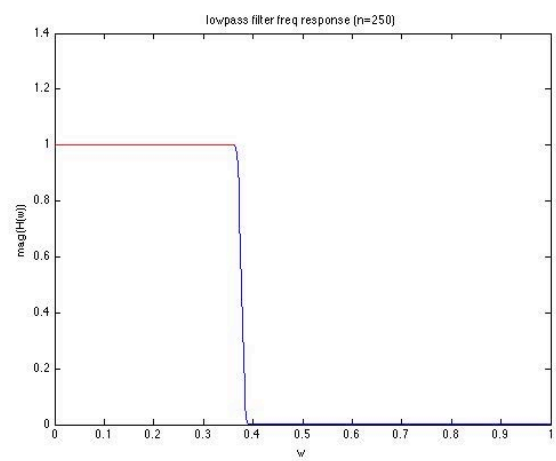




>>b=firpm(100,[0 0.3636 0.39 1],[1 1 0 0]);



n=50



n=250

To choose, you need to decide/trade off max allowable width of transition band, ripple in pass and stop bands, and how much computational power you have.

Filter Design

IIR Filters Design Methods

Description	Matlab functions
Analog Prototyping Using the poles and zeros of a classical lowpass prototype filter in the continuous (Laplace) domain, obtain a digital filter through frequency transformation and filter discretization.	butter, cheby1, cheby2, ellip
Direct Design Design digital filter directly in the discrete time-domain by approximating a piecewise linear magnitude response	yulewalk

fdatool

BUTTER Butterworth digital and analog filter design.

[B,A] = BUTTER(N,Wn)

Nth order lowpass digital Butterworth filter

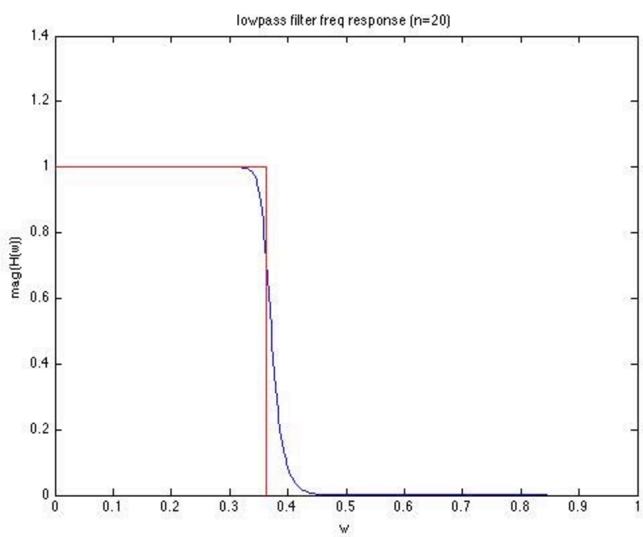
Cutoff frequency Wn 0.0 < Wn < 1.0, 1.0=half sampling rate

[B,A] = BUTTER(N,Wn,'high') designs a highpass filter.

[B,A] = BUTTER(N,Wn,'low') designs a lowpass filter.

[B,A] = BUTTER(N,Wn,’bandpass’) is a bandpass filter if Wn = [W1 W2].

[B,A] = BUTTER(N,Wn,'stop') is a bandstop filter if Wn = [W1 W2].



[b,a]=butter(20,0.3636,'low');

no ripple in pass band or stop band

wide transition band

CHEBY1 Chebyshev Type I digital design.

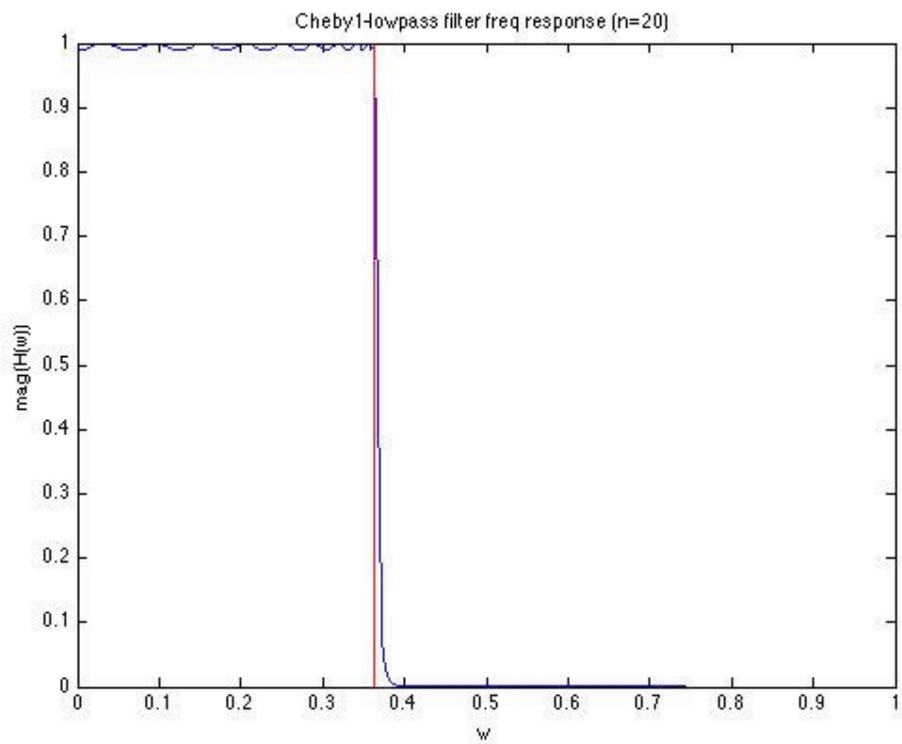
[B,A] = CHEBY1(N,R,Wp)

Nth order lowpass digital Chebyshev filter

R decibels of peak-to-peak ripple in the passband.

Wp passband-edge frequency 0.0 < Wp < 1.0,

Use R=0.5 as a starting point, if you are unsure about choosing R.



[b,a]=cheby1(20,.1,0.3636);

ripple in pass band

no ripple stop band

narrow transition band

max p-p ripple 0.1dB

$0.1dB=20\log_{10}(X/1)$

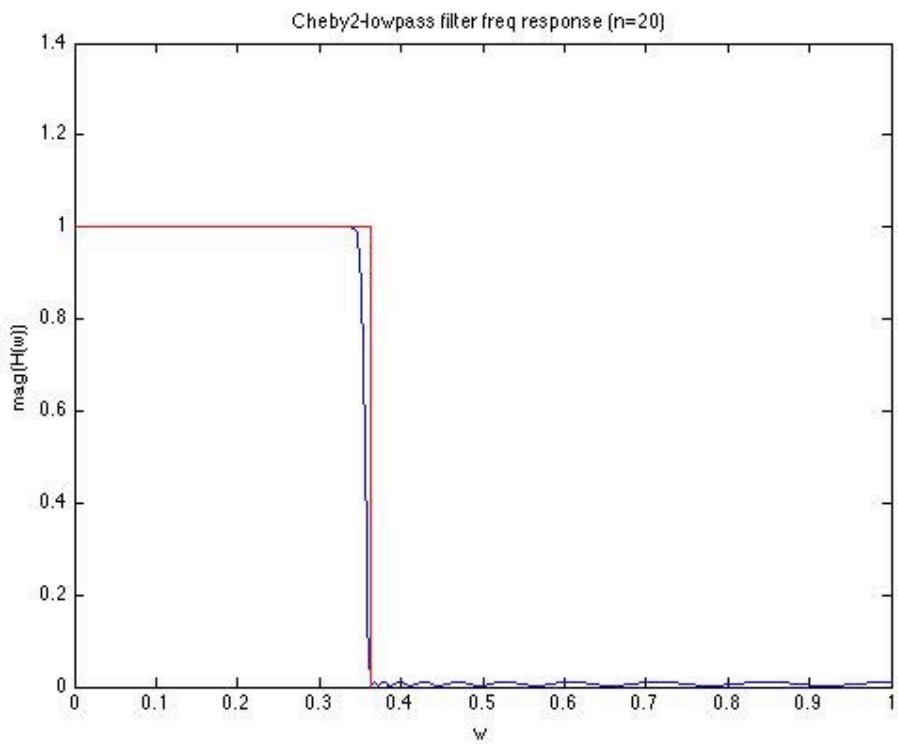
$X=10^{(0.1/20)}=1.01$

$-0.1dB=20\log_{10}(X/1)$

$X=10^{(-0.1/20)}=0.99$

CHEBY2 Chebyshev Type II digital filter design.
[B,A] = CHEBY2(N,R,Wst)
Nth order lowpass digital Chebyshev
stopband ripple R decibels down
stopband-edge frequency Wst.

Use R = 20 as a starting point, if you are unsure about choosing R.

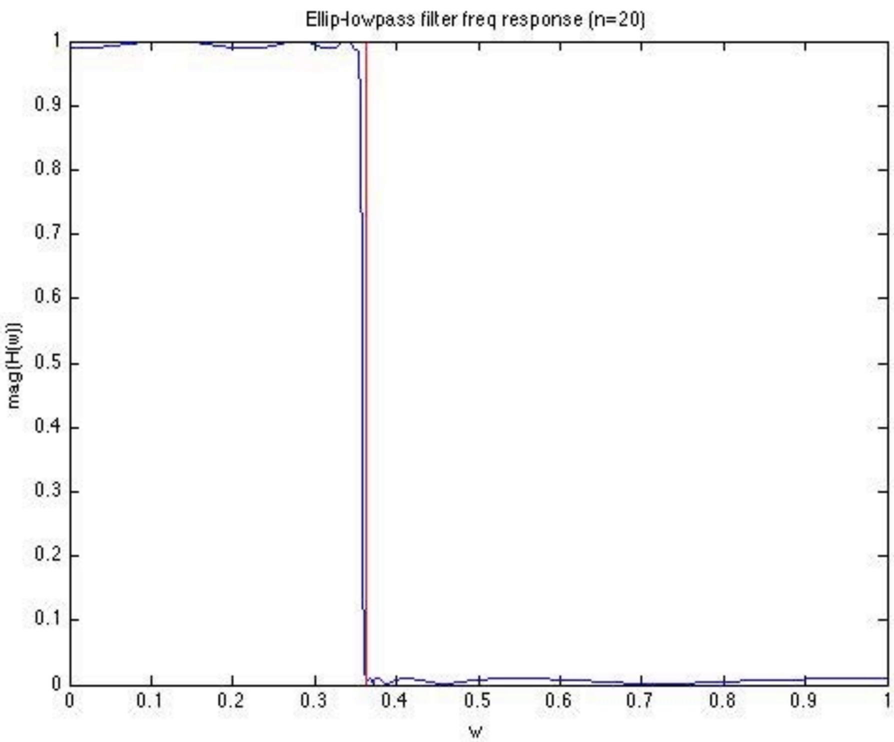


no ripple in pass band
ripple in stop band
narrow transition band

max ripple 40dB down
$-40\text{dB}=20\log_{10}(X/1)$
$X=10^{(-40/20)}=10^{-2}$
$=0.01$

[b,a]=cheby2(20,40,0.3636);

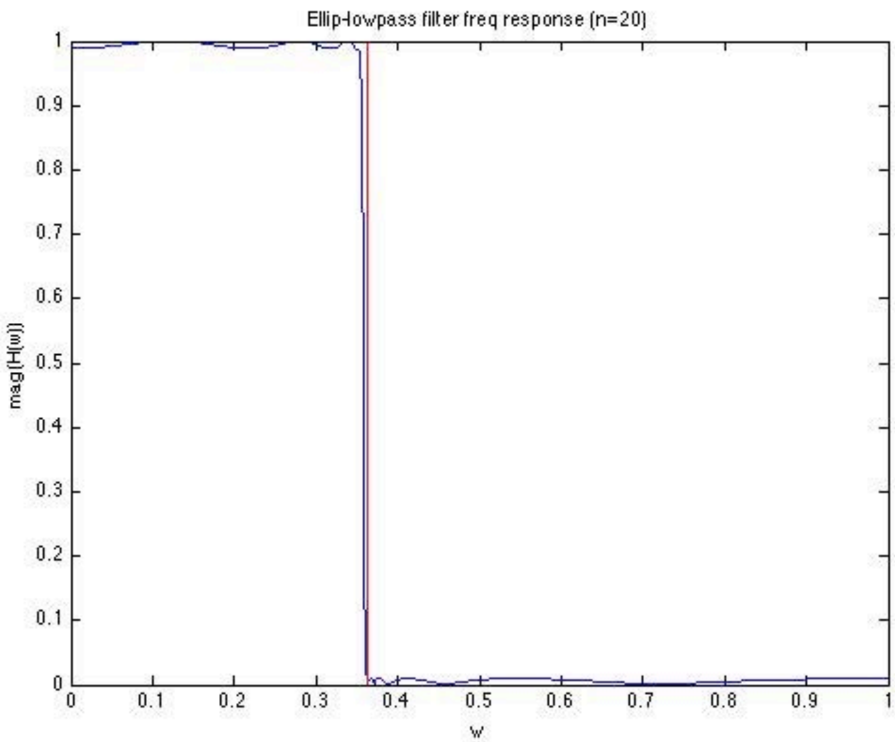
ELLIP Elliptic or Cauer digital filter design.
[B,A] = ELLIP(N,Rp,Rs,Wp
Nth order lowpass digital elliptic filter
Rp decibels of peak-to-peak ripple
Rs decibels minimum stopband attenuation
Wp passband-edge frequency 0.0 < Wp < 1.0
Use Rp = 0.5 and Rs = 20 as starting points, if you are unsure about choosing them.

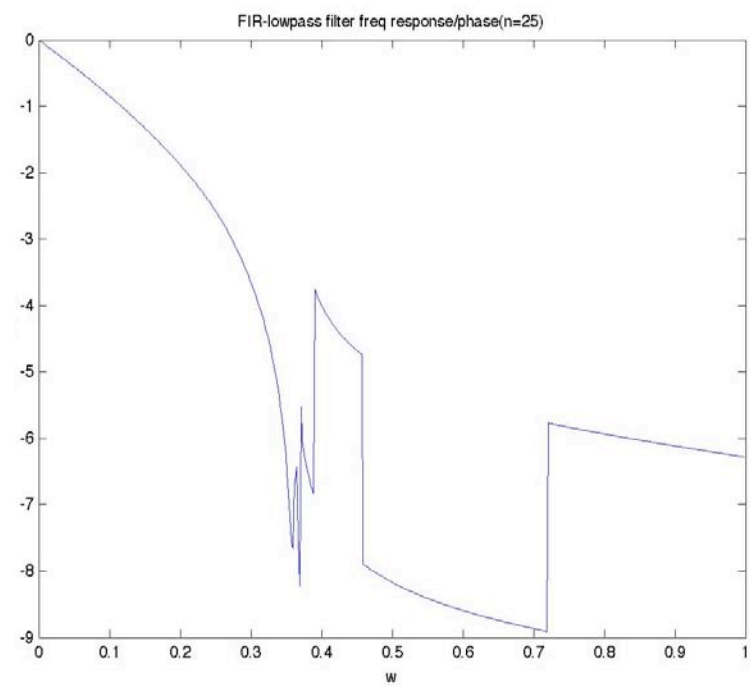


ripple in pass band
ripple in stop band
narrowest transition band

[b,a]=ellip(20,0.1,40,0.3636);

ELLIP Elliptic or Cauer digital filter design.
[B,A] = ELLIP(N,Rp,Rs,Wp
Nth order lowpass digital elliptic filter
Rp decibels of peak-to-peak ripple
Rs decibels minimum stopband attenuation
Wp passband-edge frequency 0.0 < Wp < 1.0
Use Rp = 0.5 and Rs = 20 as starting points, if you are unsure about choosing them.





[b,a]=ellip(20,0.1,40,0.3636);

IIR typically have nonlinear phase

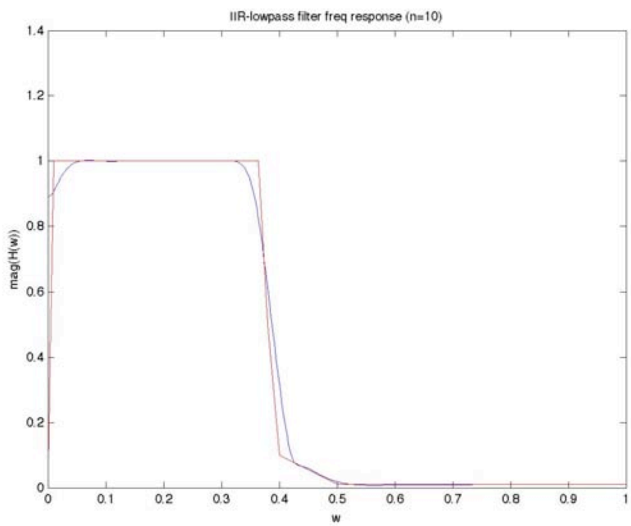
yulewalk designs IIR digital filters using a least-squares fit to a specified frequency response.

[b,a] = yulewalk(n,f,m)
N order IIR filter whose frequency-magnitude characteristics approx. match those given in vectors f and m:
f is a vector of frequency points, specified in the range between 0 and 1
The first point of f must be 0 and the last point 1, with all intermediate points in increasing order.
Duplicate frequency points are allowed, corresponding to steps in the frequency response.

m is a vector containing the desired magnitude response at the points specified in f.
f and m must be the same length.
plot(f,m) displays the filter shape.

When specifying the frequency response, avoid excessively sharp transitions from passband to stopband.
You may need to experiment with the slope of the transition region to get the best filter design.

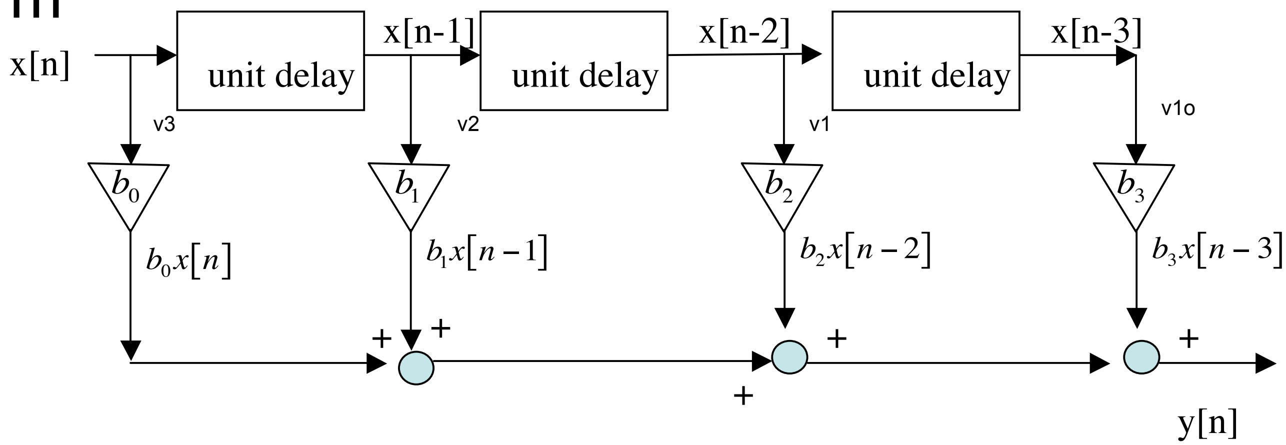
[b,a]=yulewalk(10,[0 0.005 .01 0.3 0.3636 .38 .4 .5 1],[.01 0.5 1 1. 1 .5 0.1 .01 0.01]);



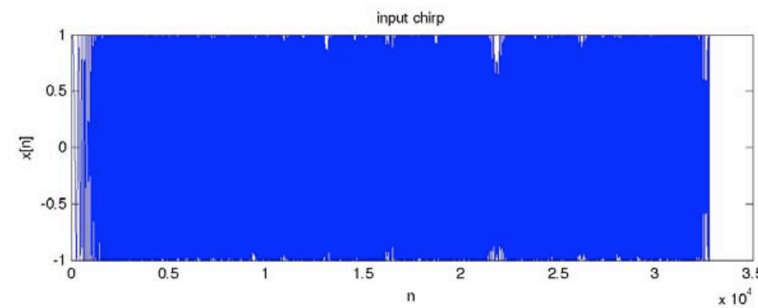
Screenshot of MATLAB FDAtool removed due to copyright restrictions.

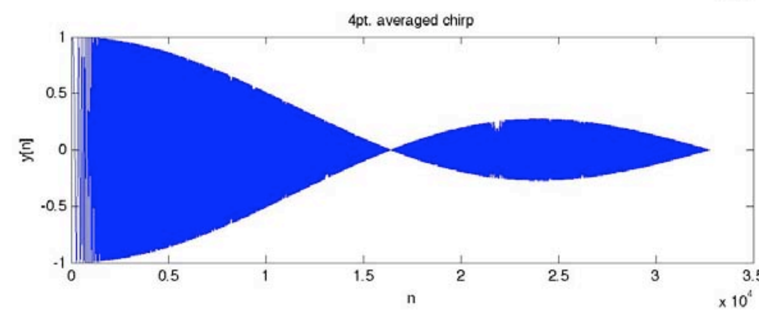
iterative implementation (FIR) direct form

$$y[n] = b_0 x[n] + b_1 x[n-1] + b_2 x[n-2] + b_3 x[n-3]$$




```
% setup input
fs=8192;
t=0:1/fs:4;
x=cos(2*pi*(8192/2/4)/2*t.^2);
%%%%%%%
% filter coefficients (4 pt averager)
b0=0.25;b1=0.25;b2=0.25;b3=0.25;

%direct form
v3o=0; %setup storage for past inputs
v2o=0;
v1o=0;

for i=1:length(x); %this would be an infinite loop for uC
   v3=x(i);   %take a sample (A/D) x[n]
   v2=v3o;   % recall  input 1 samples ago x[n-1]
   v1=v2o;   % recall  input 2 samples ago x[n-2]

% calculate output
   y1(i)=b0*v3+b1*v2+b2*v1+b3*v1o;

% store inputs
v1o=v1; % save input 2 sample ago x[n-3]=x[n-2]
v2o=v2; % save input 1 sample ago  x[n-2]=x[n-1]
v3o=v3; % save current input  x[n-1]=x[n]

end
```

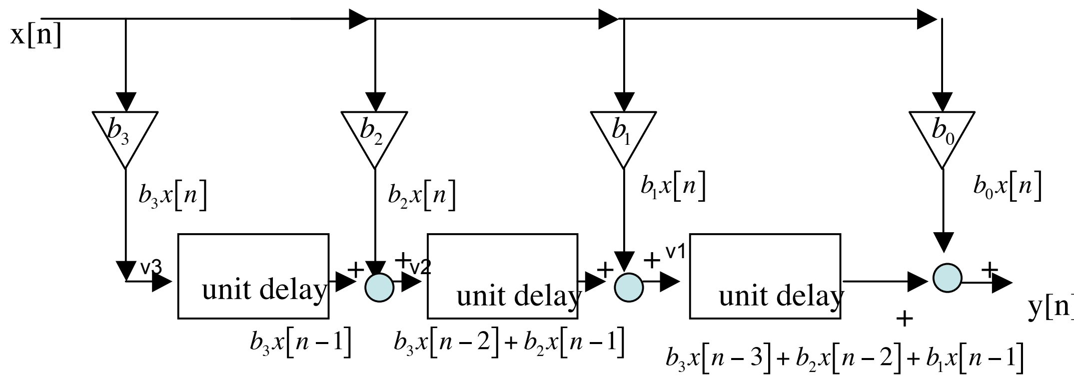




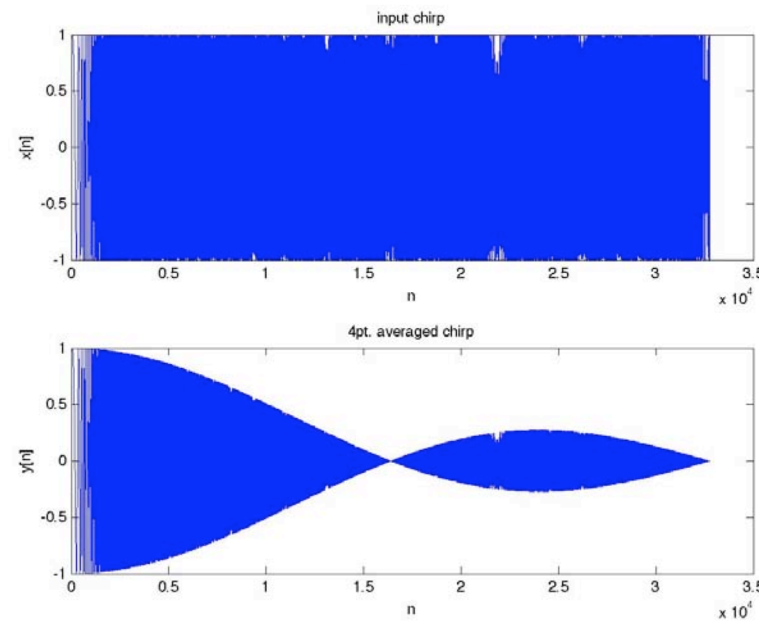
Optimize
Use better structures
Add feedforward (IIR)
(store outputs and add to difference eqn)

iterative implementation (FIR) transpose form

$$y[n] = b_0 x[n] + b_1 x[n-1] + b_2 x[n-2] + b_3 x[n-3]$$

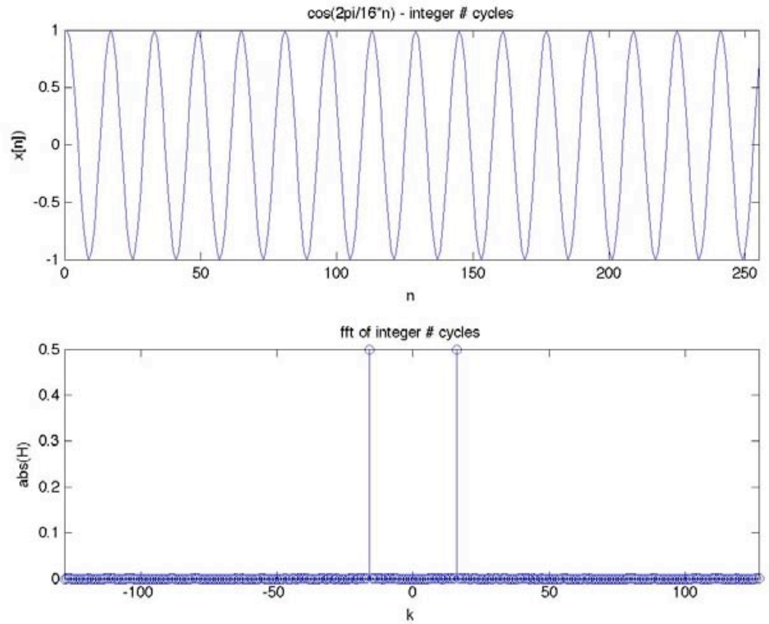



```
% setup input
fs=8192;
t=0:1/fs:4;
x=cos(2*pi*(8192/2/4)/2*t.^2);
%%%%%%%
% filter coefficients (4 pt averager)
b0=0.25;b1=0.25;b2=0.25;b3=0.25;

%transpose form
v3=0; %setup storage for past combos
v2=0;
v1=0;

for i=1:length(x); %this would be an infinite loop for uC

% calculate output
   xs=x(i);  %take a sample (A/D)
   y2(i)=b0*xs+v1; %b0*x[n]+(b1*x[n-1]+b2*x[n-2]+b3*x[n-3])

% store combos
   v1=b1*xs+v2; %b1*x[n]+(b2*x[n-1]+x3*x[n-2])
   v2=b2*xs+v3; %b2*x[n]+x3*x[n-1]
   v3=b3*xs; %b3*x[n]
end
```



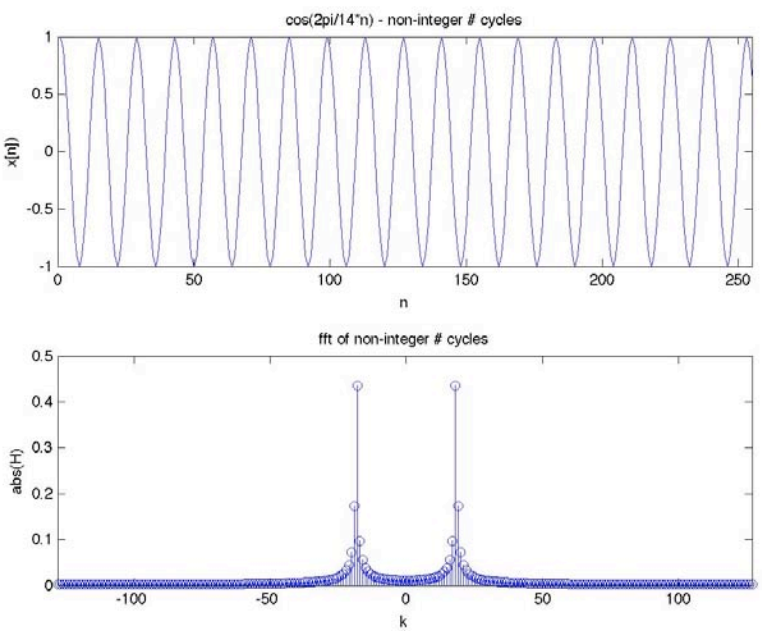
Direct Form versus Transposed Form implementations; in the former, input samples are buffered (i.e. effectively move through a delay line) whereas in the Transpose version, partial sums are stored and propagated. Although the theoretical number of computations is often nominally the same, these differences will often show up in word lengths required, control logic, pipeline stages, etc. -http://syndicated.synplicity.com/Q207/dsp.html

FFT Windows



Integer # of cycles

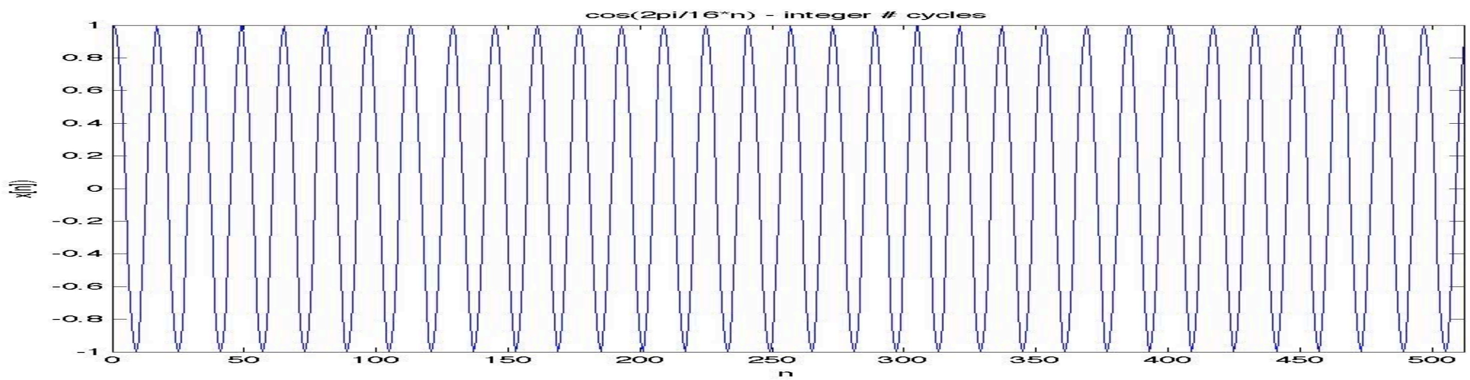
n=0:255
x=cos(2*pi/16*n)
y=fftshift(abs(fft(x))/256);

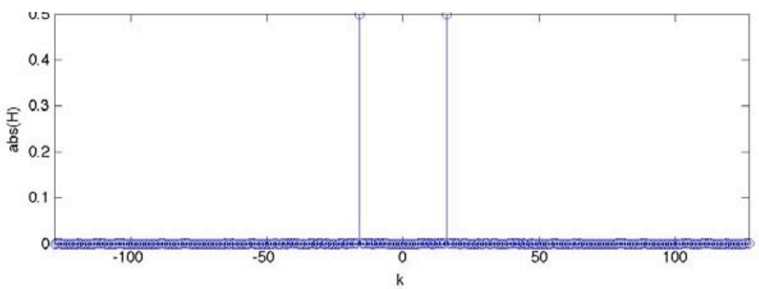


Non-integer # of cycles

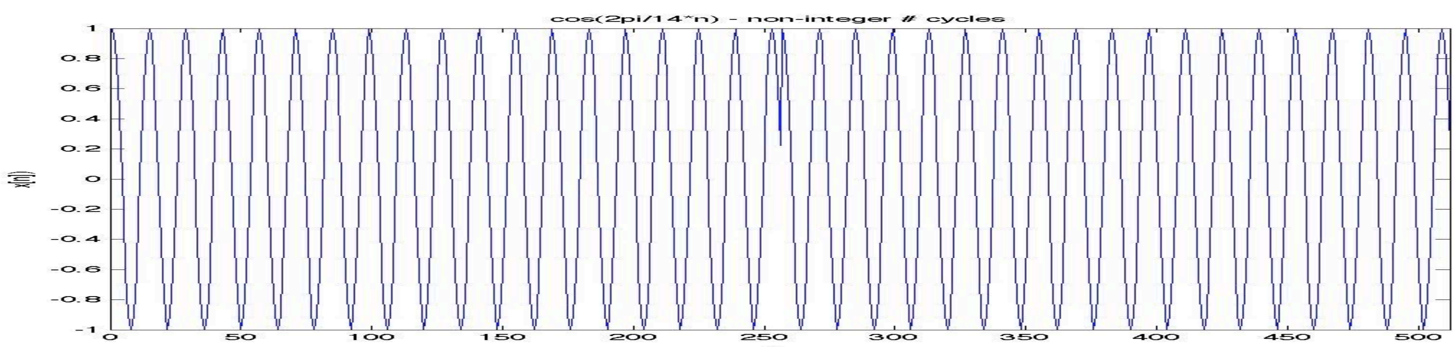
n=0:255
x=cos(2*pi/14*n)
y=fftshift(abs(fft(x))/256);

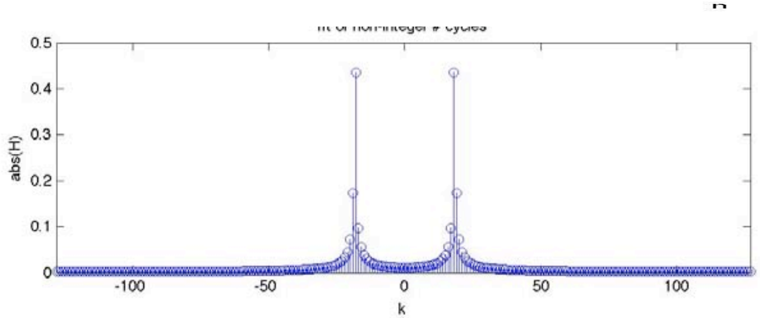
Note: fft(x) for freq response
fft(x)/L for spectrum



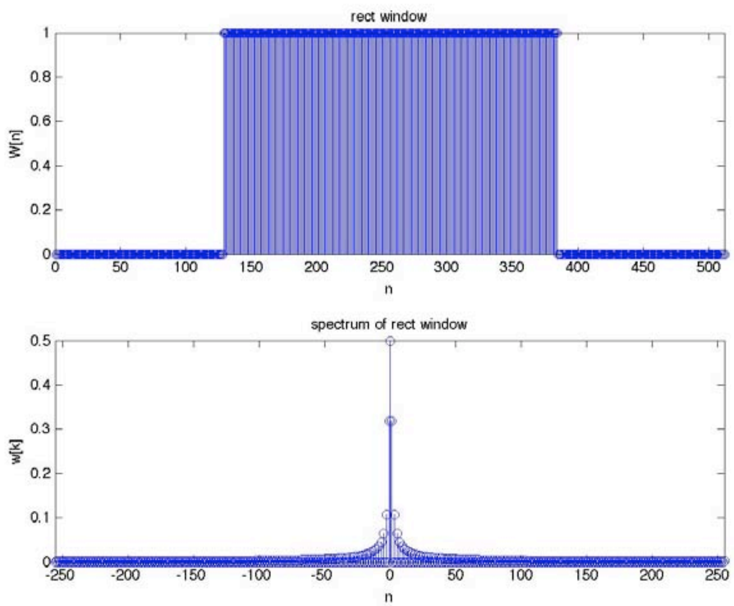


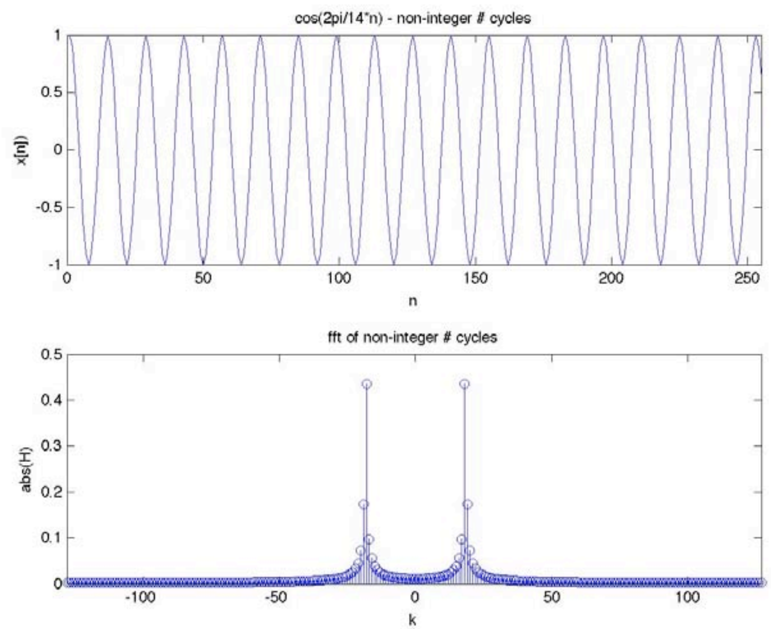
Integer # of cycles
Continuous/periodic





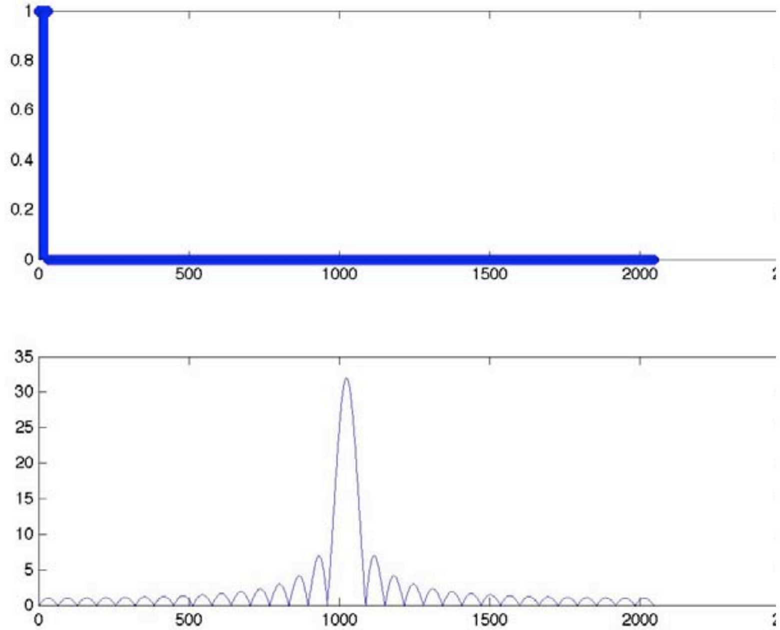
Non-integer # of cycles
Discontinuity has lots of freq.
leakage



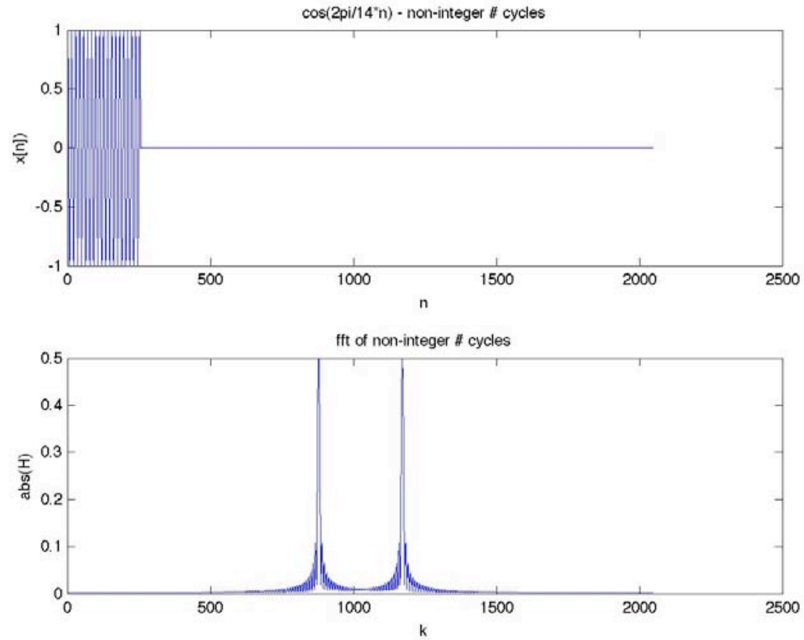


Sampled time frame equivalent to multiplying signal by a rectangular (boxcar) window.

Spectrum equivalent to convolving periodic signal's spectrum (spikes) by a spectrum rectangular window (sinc-like). Sinc's sidelobes pick up extraneous frequency contributions.

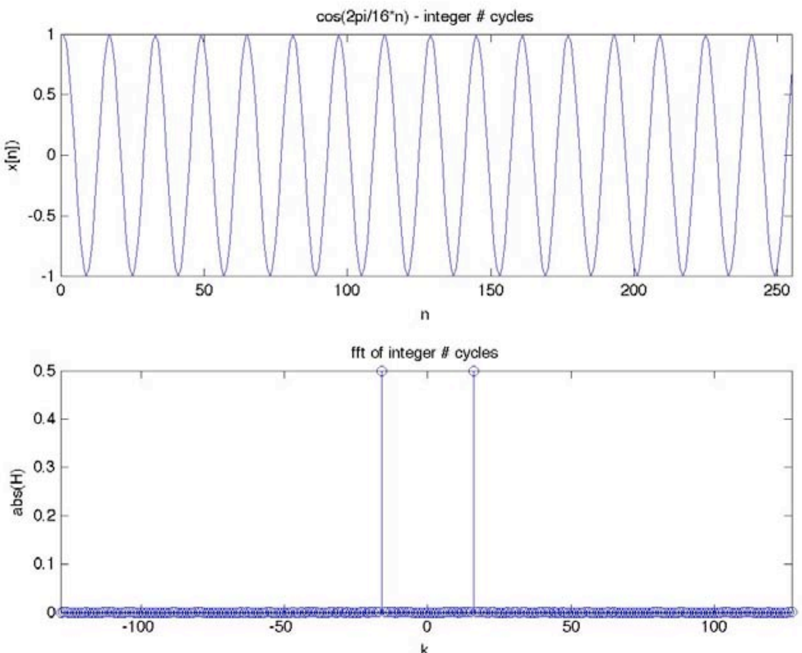


Sampled time frame equivalent to multiplying signal by a rectangular (boxcar) window.

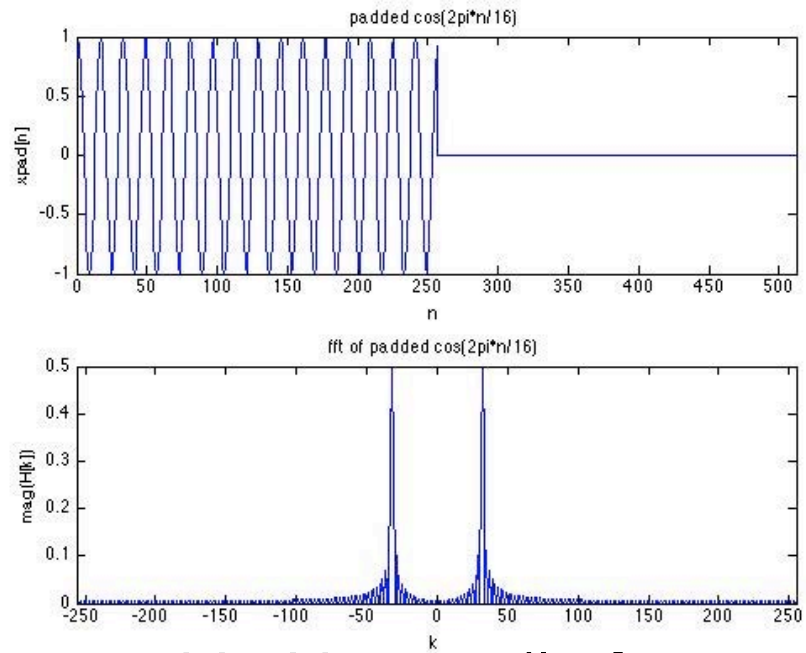


Spectrum equivalent to convolving periodic signal’s spectrum (spikes) by a spectrum rectangular window (sinc-like). Sinc’s sidelobes pick up extraneous frequency contributions.

Padding is also windowing



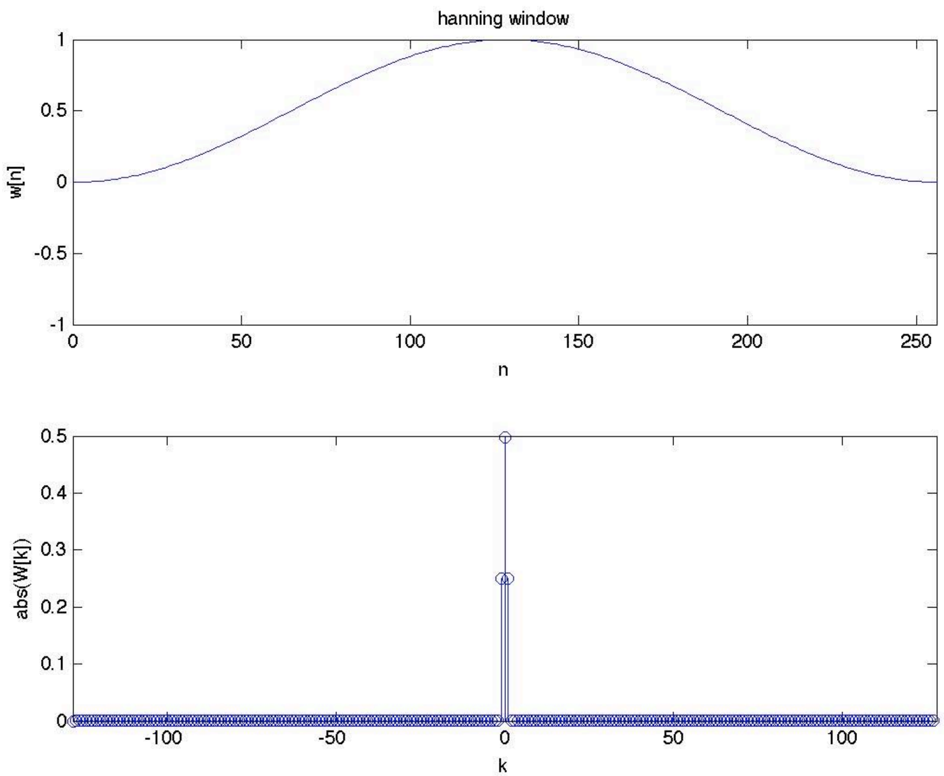
Integer # of cycles
Continuous/periodic



padded integer# of cycles
discontinuous

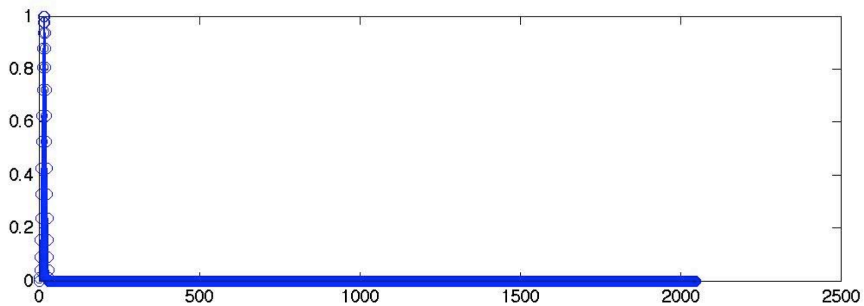
Padding adds more bins (frequency samples), but introduces leakage, and doesn't increase real resolution
(resolution = highest freq. $\hat{\omega} = \pi$
still defined by sampling rate, not record length)

Soln: Use other windows to make finite sampled signal look periodic and continuous in time frame.
Try to reduce window's sidelobe's to reduce picking up other frequency contributions.

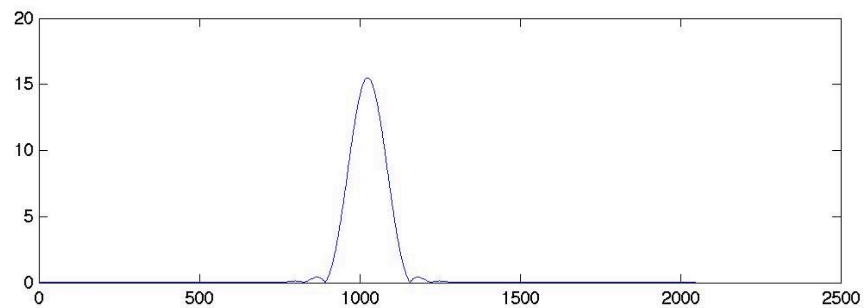


Hanning window

Soln: Use other windows to make finite sampled signal look periodic and continuous in time frame.
Try to reduce window's sidelobes to reduce picking up other frequency contributions.

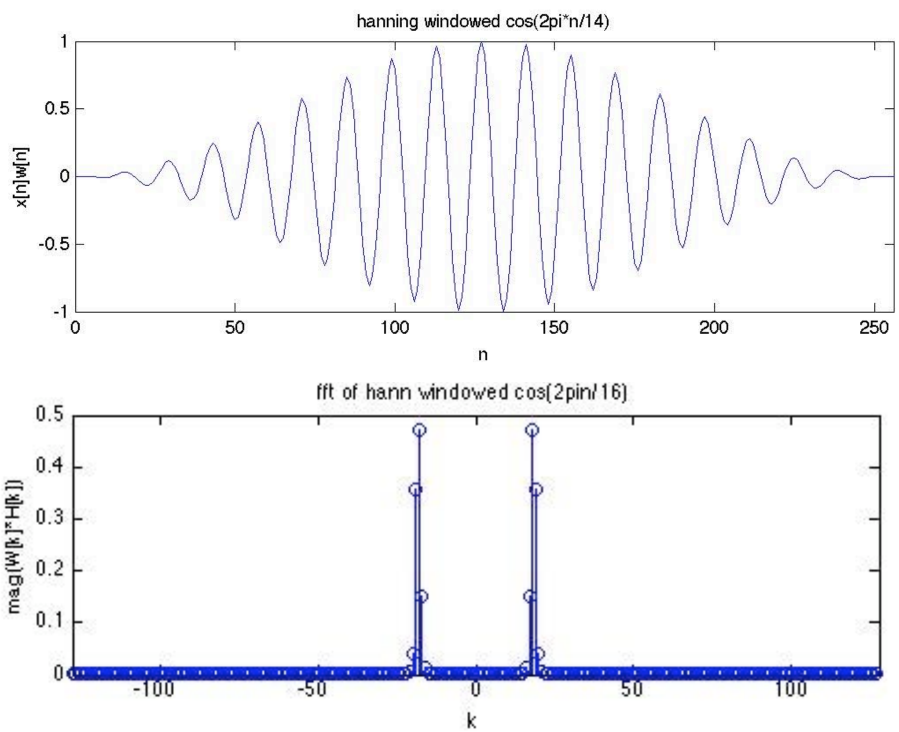


Hanning window



wider mainlobe than boxcar
But smaller sidelobes

Soln: Use other windows to make finite sampled signal look periodic and continuous in time frame.
Try to reduce window's sidelobes to reduce picking up other frequency contributions.

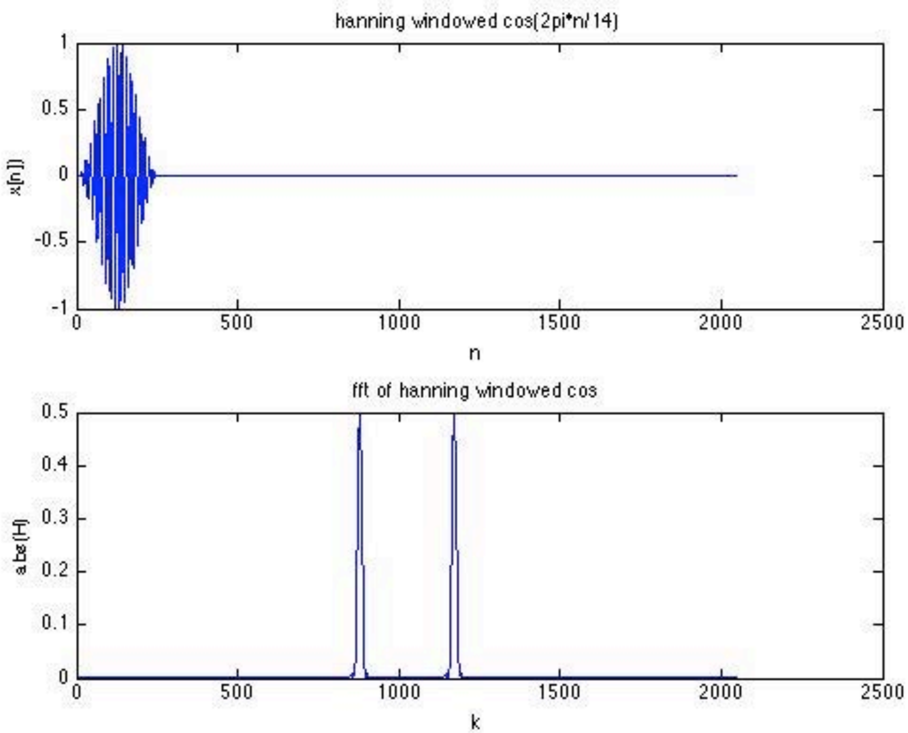


Hanning windowed cosine
Now signal looks periodic
in time frame

```
w2=window(@hann,256);
x=cos(2*pi/14*n);
x2=x.*w2';
y=fftshift(abs(fft(x2))*2/256))
```

need to account for window attenuation

Soln: Use other windows to make finite sampled signal look periodic and continuous in time frame.
Try to reduce window's sidelobe's to reduce picking up other frequency contributions.

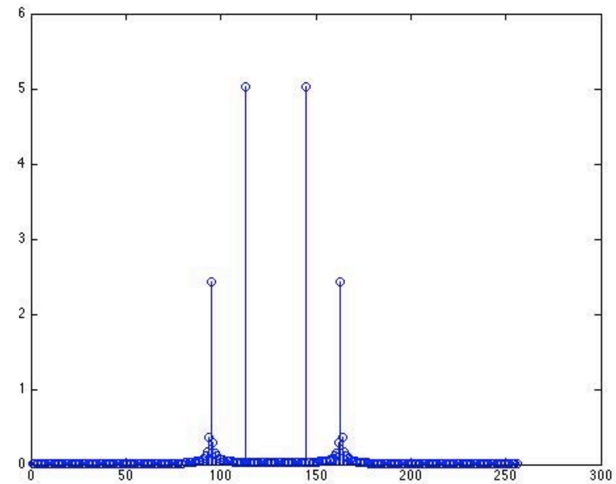


Hanning windowed cosine
Now signal looks periodic
in time frame

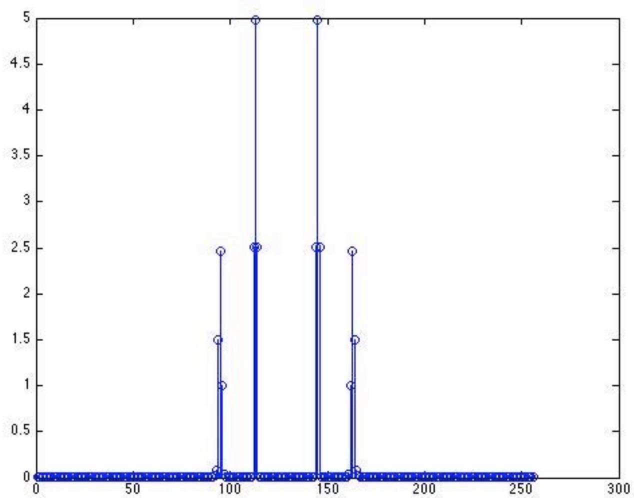
less frequency resolution
(wider main lobe)

but better amplitude (?)
(smaller side lobes)

$$x[n] = 10\cos(2\pi/16 \cdot n) + 5\cos(2\pi/7.5 \cdot n)$$

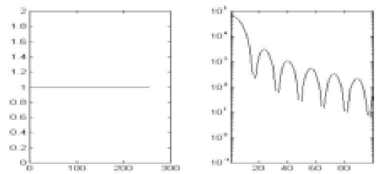
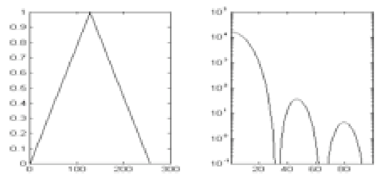
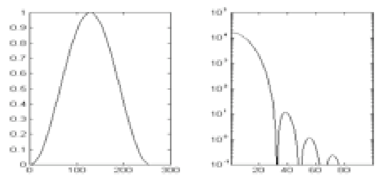
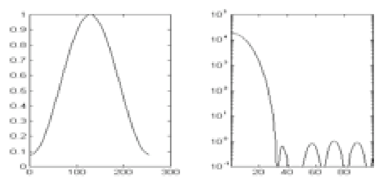
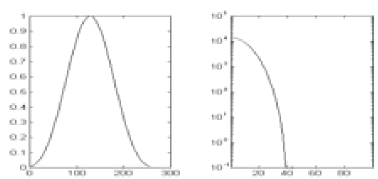


Boxcar
greater frequency resolution
(smaller main lobe)
but worse amplitude
(smaller side lobes)



Hanning
less frequency resolution
(wider main lobe)
but better amplitude
(smaller side lobes)

Table .1: Some often used window functions

Window (Matlab)	Description w[k]=..., k=1..N	
Boxcar	rectangular (=1)	
Triang	$\begin{cases} \dfrac{2k-1}{N}, & 1 \le \mathrm{k} \le \dfrac{\mathrm{N}}{2} \\ \dfrac{2(N-k+1)}{N}, & \dfrac{\mathrm{N}}{2}+1 \le k \le N \end{cases}$	
Hanning	$0.5\left(1-\cos 2\pi \dfrac{k}{N+1}\right)$	
Hamming	$0.54 - 0.46 \cdot \cos 2\pi \dfrac{k-1}{N-1}$	
Kaiser	Spherical Bessel function with parameter β . β =0 Þ "rectangular window" larger β Þ better side lobe reduction but broader lines. Here β =7.	

Trade off between frequency resolution (main lobe width) and leakage (side lobes)

Text removed due to copyright restrictions. Description of advantages and preferred applications of common windowing types.
From National Semiconductor website "Windowing: Optimizing FFTs Using Window Functions," http://zone.ni.com/devzone/cda/tut/p/id/4844.

WINDOW(@WNAME,N) returns an N-point window of type specified by the function handle @WNAME in a column vector. @WNAME can be any valid window function name, for example:

```
@bartlett       - Bartlett window.
@barthannwin    - Modified Bartlett-Hanning window.
@blackman       - Blackman window.
@blackmanharris - Minimum 4-term Blackman-Harris window.
@bohmanwin      - Bohman window.
@chebwin        - Chebyshev window.
@flattopwin     - Flat Top window.
@gausswin       - Gaussian window.
@hamming        - Hamming window.
@hann           - Hann window.
@kaiser         - Kaiser window.
@nuttallwin     - Nuttall defined minimum 4-term Blackman-Harris window.
@parzenwin      - Parzen (de la Valle-Poussin) window.
@rectwin        - Rectangular window.
@tukeywin       - Tukey window.
@triang         - Triangular window.
```

Cepstrum

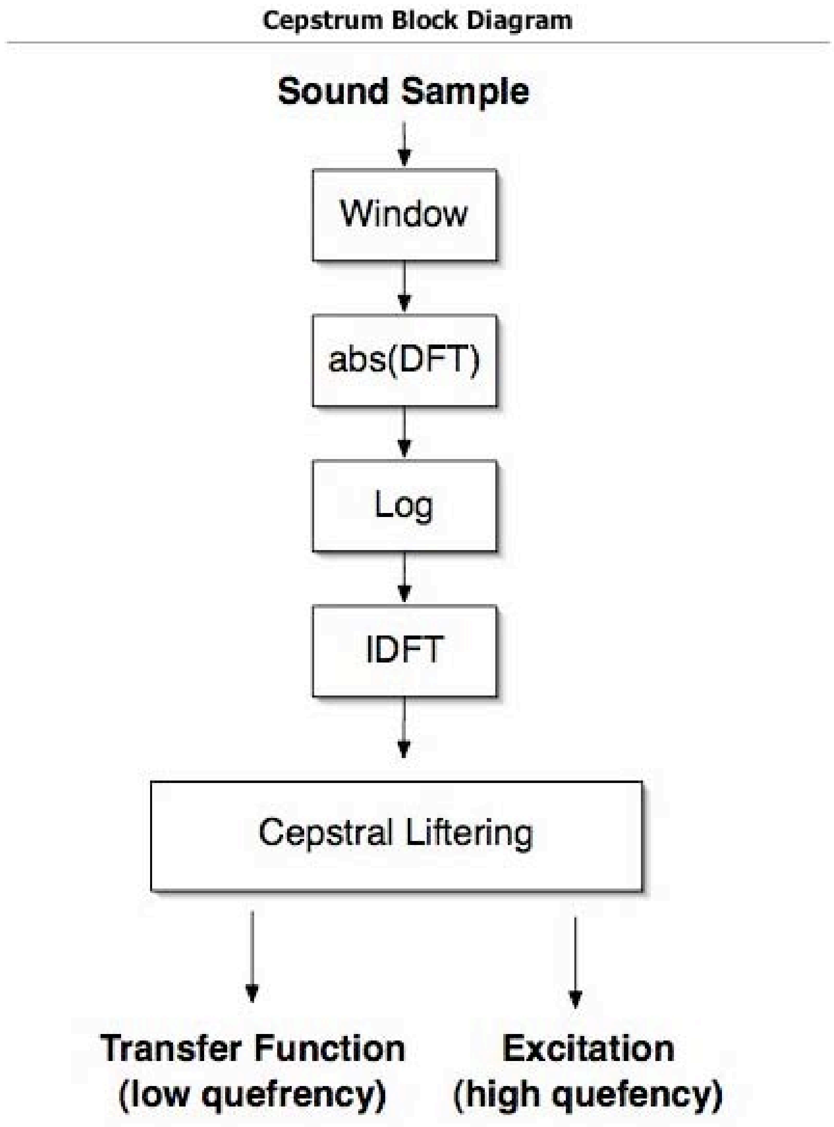
Voice = vocal tract * periodic excitation (vocal cords))

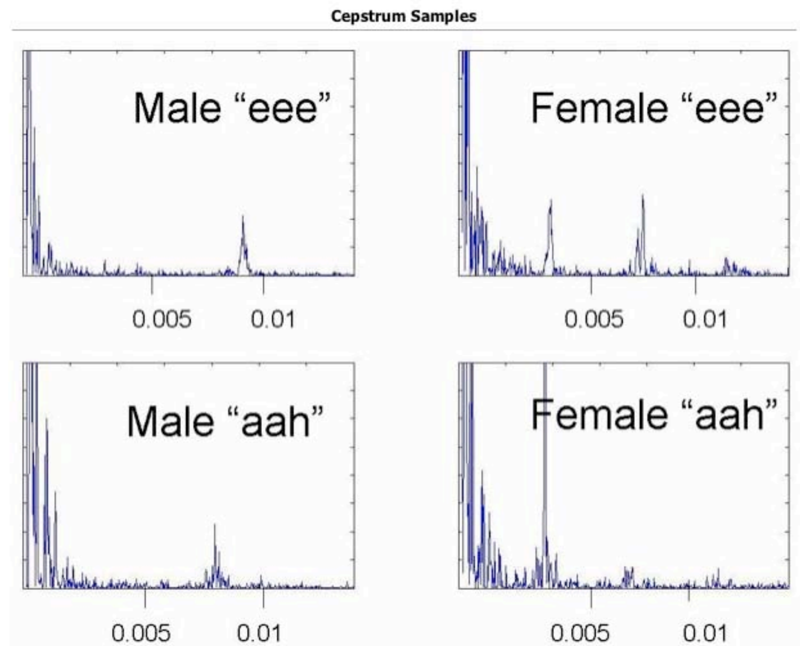
y[n]=h[n]*x[n]
fft
Y[n]=H[n]X[n]

log(Y[k])=log(H[k])+log(X[k])
low freq high freq
ifft
quefrencies

Separate out system and signal. (remove user effects)

Uses homographic filtering (use log to separate product
and fft to look at slow and fast variations, then filter)
(homographic filtering also used to separate albedo vs. lighting)





The transfer function usually appears as a steep slant at the beginning of the plot.
The excitation appears as periodic peaks occurring after around 5ms.
The female voice has peaks occurring more often then in the male’s cepstrum. This is due to the higher pitch of a female voice.

http://cnx.org/content/m12469/latest/