Signal Processing in MATLAB

February 2, 1998

Tom Krauss

PhD Student in Elec. Engineering

On-campus Representative of The MathWorks, Inc.

krauss@ecn.purdue.edu

Outline

- Built-in MATLAB support for Signal Processing
- Overview of the Signal Processing Toolbox
- Some new MATLAB 5 Signal Processing features

How to Start MATLAB

- Login to UNIX prompt
- type

    matlab

Signal Representation

- Weighted Sum of Sinusoids

  x[n] = a1*sin(w1*n+phi1) + a2*sin(w2*n+phi2) +
  a3*sin(w3*n+phi3), 0 <= n <= N-1

- Implementation 1 - non-vectorized, ala C or FORTRAN

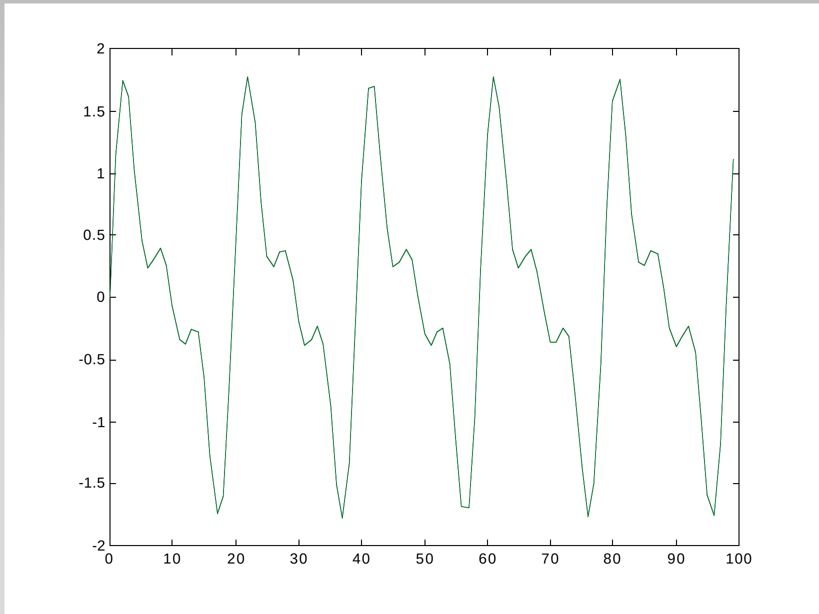
```
N = 100;
a = [1 1/sqrt(2) 0.5];
w = [1 2 3]*.051*2*pi;
phi = [0 0 0];

x = zeros(N,1);
for n = 0:N-1
    for k = 1:3
       x(n+1) = x(n+1) + a(k)*sin(w(k)*n + phi(k));
    end
end
```

Vectorization

- Implementation 2 - vectorized

```
n = 0:N-1;
x1 = sin(n'*w + phi(ones(1,N),:))*a';
```

- Plotting and Comparison
  - plot(0:N-1,[x x1])
  - norm(x-x1)



How did we do that?

```
sin(n'*w + phi(ones(1,N),:))*a'
```

- dash ’ is the (conjugate) transpose of a matrix
- Outer-product n’*w is N-by-3 matrix - think of it as a weighted replication of row w with elements of n as weights
- Replicated row phi into matrix [phi; phi; ... phi] using : indexing notation
- Sum and sine are *element-wise* operations
- Matrix Multiplication is a linear combination of column vectors

```
               w1*0 + phi1       w2*0 + phi2       w3*0 + phi3
               w1*1 + phi1       w2*1 + phi2       w3*1 + phi3
               w1*2 + phi1       w2*2 + phi2       w3*2 + phi3
x[n] = sin (   w1*3 + phi1       w2*3 + phi2       w3*3 + phi3    ) * a'
               .                  .                  .
               .                  .                  .
               .                  .                  .
               w1*(N-2)+phi1   w2*(N-2)+phi2   w3*(N-2)+phi3
               w1*(N-1)+phi1   w2*(N-1)+phi2   w3*(N-1)+phi3
```

Functions - sos.m

- Create a file entitled sos.m containing the following text:

```
function x = sos(N,a,w,phi)
% SOS Weighted sum of sinusoids
% Inputs:
%  N - length of sequence
%  a - vector of amplitudes
%  w - vector of frequencies (in radians)
%  phi - vector of phases
% Uses Implementation 1 (non vectorized)

for n = 0:N-1
    x(n+1) = 0;
    for k = 1:3
       x(n+1) = x(n+1) + a(k)*cos(w(k)*n + phi(k));
    end
end
```

About Functions

- Help is automatic
  - The help for the function is everything after the ‘function’ line that starts with % (the comment character), up to the first line that is not a comment.
  - To get help on our function, (or ANY function in MATLAB), type ‘help sos.m’
- Input and output arguments

```
function x = sos(N,a,w,phi)
```

  - This line defines N, a, w and phi as input arguments, and x as output argument
  - These arguments are local to the function sos
    - We can have a variable of the same name in the calling workspace
    - Don’t exist before or after function execution

Another Function - sos1.m

```
function x1 = sos1(N,a,w,phi)
% SOS1 Weighted sum of sinusoids
% Inputs:
%  N - length of sequence
%  a - vector of amplitudes
%  w - vector of frequencies (in radians)
%  phi - vector of phases
% Uses Implementation 2 (vectorized)

n = 0:N-1;
x1 = cos(n'*w + phi(ones(1,N),:))*a';
```

Timing Comparison

- Use tic and toc:

```
>> tic, for i=1:100, x=sos(N,a,w,phi); end, t=toc
t =
       2.4385

>> tic, for i=1:100, x1=sos1(N,a,w,phi); end, t1=toc
t1 =
      0.11287
>> t/t1
ans =
       21.605
```

FACTOR OF 20 SPEED INCREASE BY USING VECTORIZATION

Noise Signals

- There are two functions, rand and randn, which generate matrices of random numbers
  - x=rand(m,n) creates m-by-n matrix of independent, uniformly distributed real numbers on interval [0,1]
  - x=randn(m,n) creates m-by-n matrix of independent, Normally distributed real numbers with mean 0 and variance 1
- rand and randn have internal states that determine the numbers produced. This state is initialized when you start MATLAB, and you can reset it at will. Example:

```
>> rand(1,3)
ans =
      0.95013      0.23114      0.60684
>> rand('state',0)
>> rand(1,3)
ans =
      0.95013      0.23114      0.60684
```

More on Noise

- Note that the random number generators have been improved in MATLAB 5. The old random number generators are still present and are activated by the command rand('seed',0). It is recommended that you remove any references to 'seed' from all your old MATLAB code so that you can use the improved generators.
- Example:
  - High frequency noise

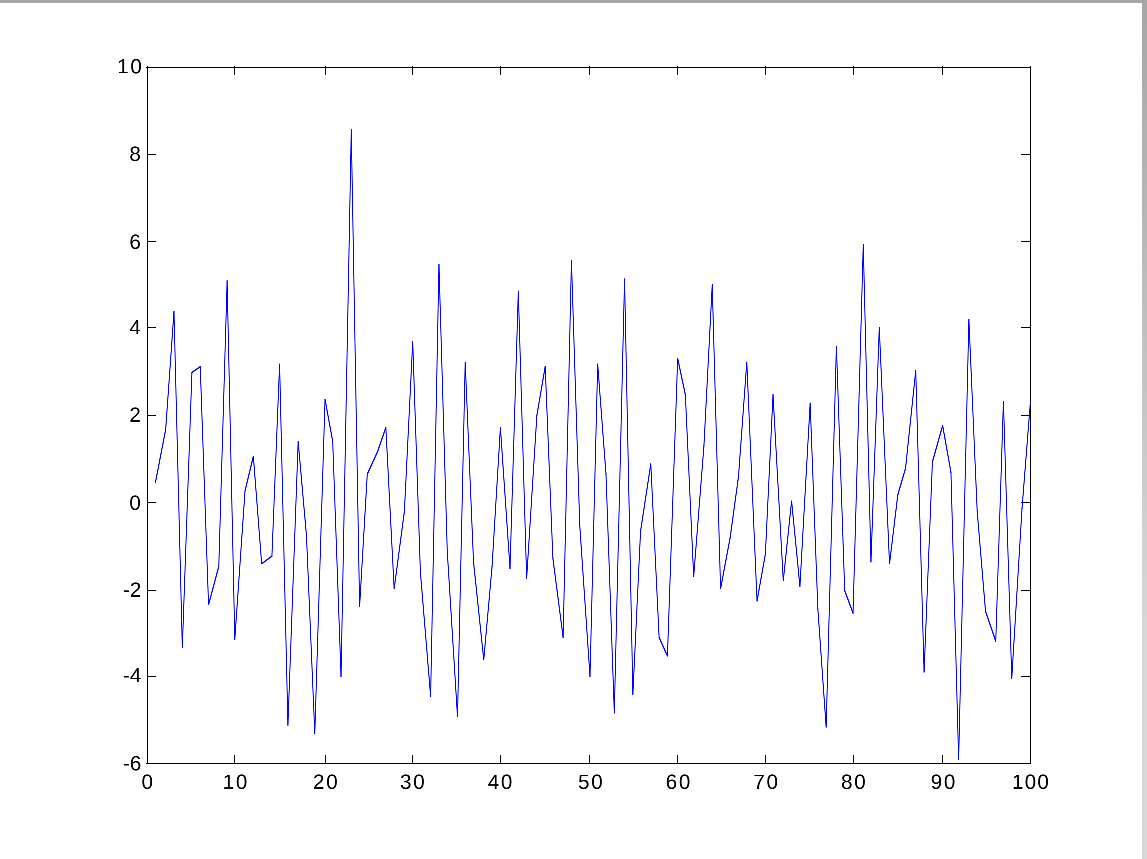
More about these functions later!

```
h = remez(40,[0 .4 .6 1],[0 0 1 1]);
noise = 5*randn(2*N,1);
noise = filter(h,1,noise);
noise = noise(end-N+1:end);  % make it length N
```

Note use of MATLAB 5 feature "end"

Additive Noise Example

```
x = x + noise;
plot(0:N-1,x)
```



The signal is completely buried!

Attempt to remove noise

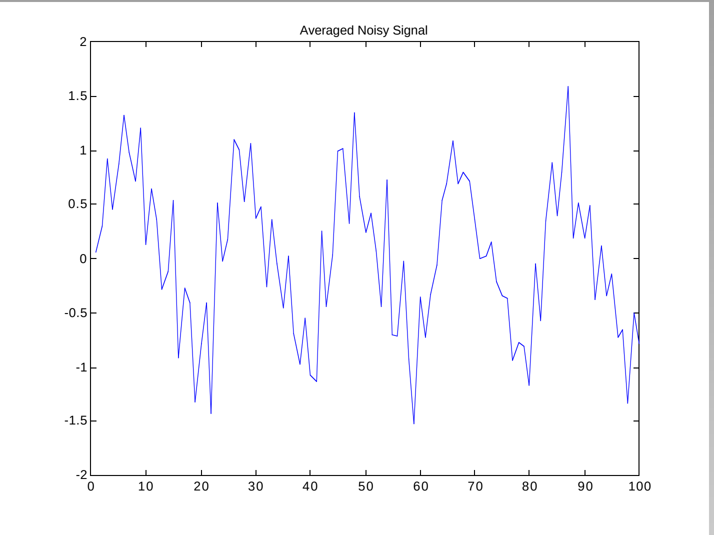
- Smoothing with running average of 7 points

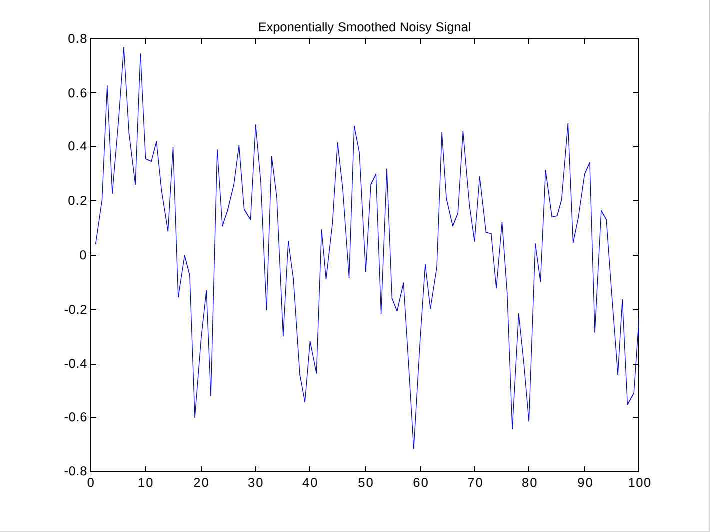
  y[n] = 1/7 * (x[n] +

  x[n-1] + ... + x[n-6])

- "Exponential Smoothing" where present sample is forced to be similar to previous sample

  y[n] = 0.9*y[n-1] + 0.1*x[n]





Digital Filtering with *filter*

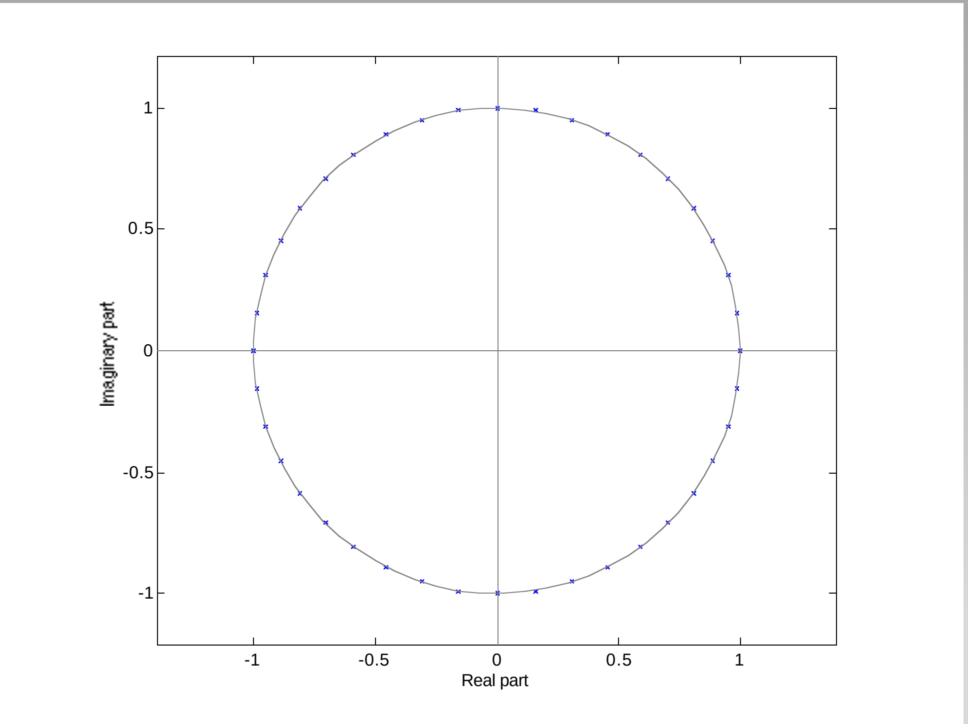
- Both these smoothing operations and more can be implemented with filter:

```
y = filter([1 1 1 1 1 1 1]/7,1,x);  % moving average FIR
y = filter(.1,[1 -.9],x);  % exponential smoothing IIR
```

```
FILTER One-dimensional digital filter.
    Y = FILTER(B,A,X) filters the data in vector X with the
    filter described by vectors A and B to create the filtered
    data Y.  The filter is a "Direct Form II Transposed"
    implementation of the standard difference equation:

    a(1)*y(n) = b(1)*x(n) + b(2)*x(n-1) + ... + b(nb+1)*x(n-nb)
                          - a(2)*y(n-1) - ... - a(na+1)*y(n-na)
```

Transforms: FFT

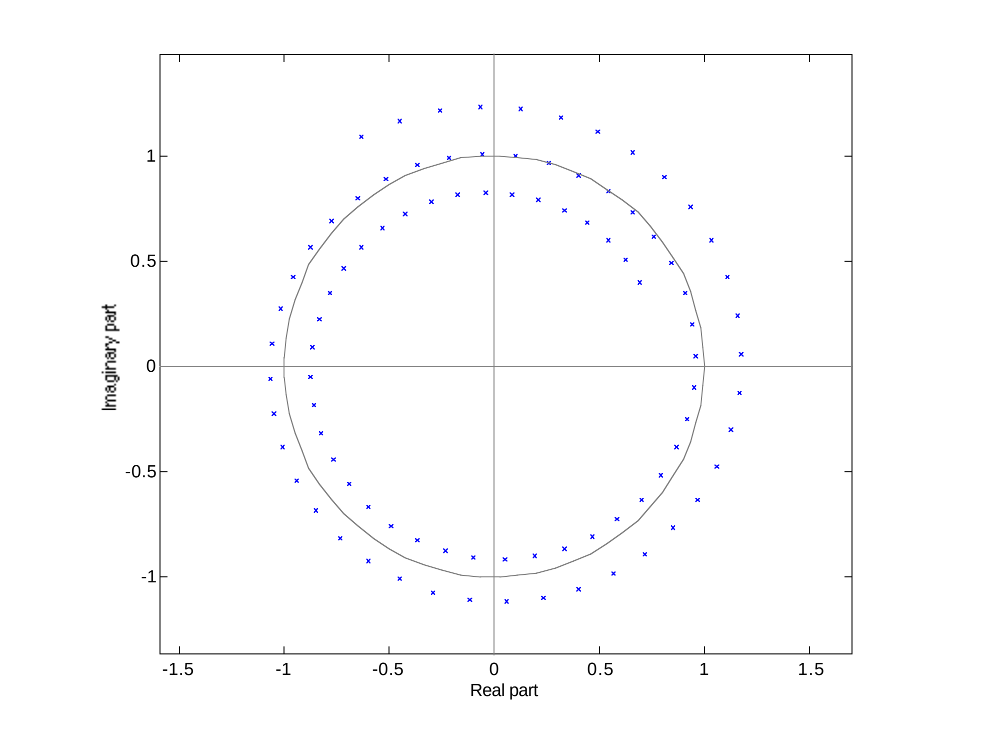
```
A = 1;
W = exp(-j*pi*.05);
M = 40;
z = A* W.^(-(0:M-1));

zplane([],z.')
```



Computes Z-transform at evenly-spaced points around the unit circle

Transforms: CZT

```
A = .8  *exp( j*pi/6);
W = .995*exp(-j*pi*.05);
M = 91;
z = A* W.^(-(0:M-1));

zplane([],z.')
```



Domain is a spiral or “chirp” in the Z-plane

FFT vs. CZT

fft(x,M) Uses prime factor algorithm which is fastest when transform length is a power of 2.

czt(x,M,W,A) Uses next greatest power–of–2 FFT for fast computation.

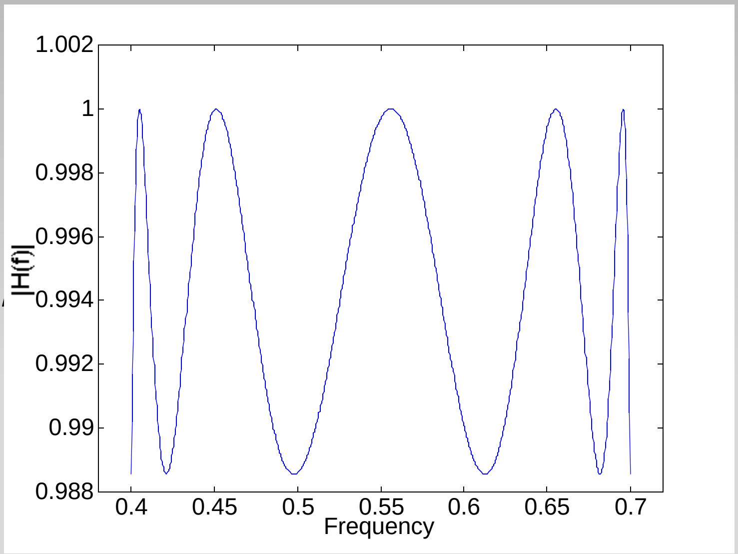
Timing Example

```
>> x=randn(2027,1);  % a prime length sequence
>> tic, fft(x); toc
elapsed_time =
      0.18934
>> tic, czt(x); toc
elapsed_time =
      0.11305
```

Note: both fft and czt work column-wise on matrices - very useful

Zoom Transform Application of Chirp Z-transform

```
f1 = .4;
f2 = .7;
[b,a] = ellip(5,.1,40,[f1 f2]);

M = 1000;
A = exp(j*pi*f1);
W = exp(-j*pi*(f2-f1)/(M-1));

H = czt(b,M,W,A)./czt(a,M,W,A)
f = linspace(f1,f2,M);

plot(f,abs(H))
```



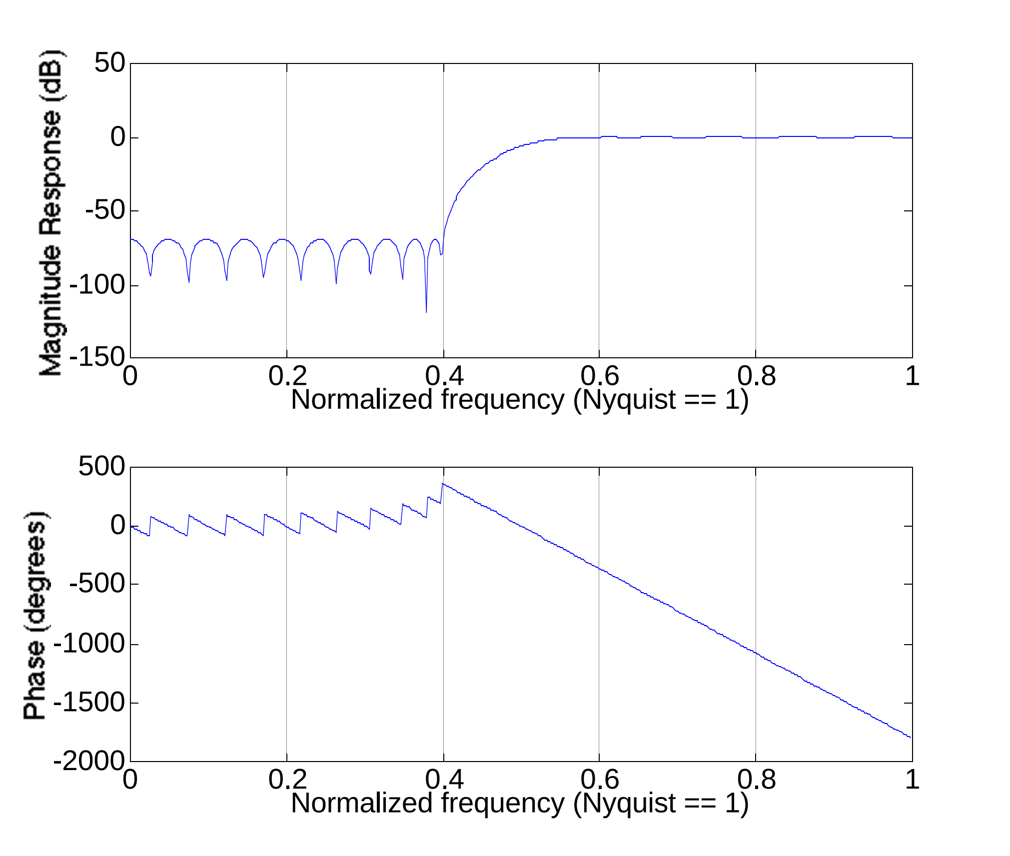
Efficiently computes frequency response in passband only

Signal Processing Toolbox Overview - FIR Filter Design

- FIR example: `h = remez(40,[0 .4 .6 1],[0 0 1 1]);`

  `freqz(h)`

High-pass Finite Impulse Response equiripple filter

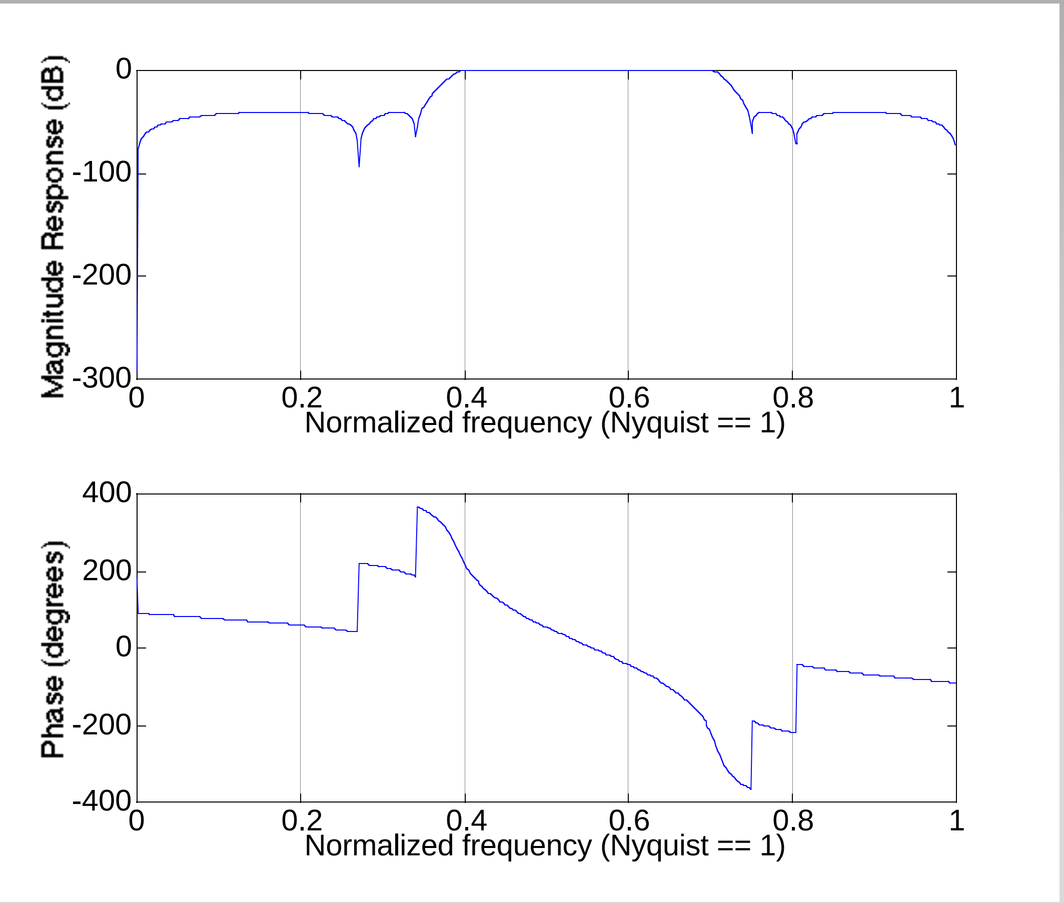


Signal Processing Toolbox Overview - IIR Filter Design

- IIR example: `[b,a] = ellip(5,.1,40,[.4 .7]);`

```
freqz(b,a)
```

Band-pass Infinite Impulse Response equiripple (elliptic) filter



Signal Processing Toolbox Overview - Other Filter Design Techniques

- FIR
  - Parks-McClellan (minimax) `remez`
  - Least Squares `firls`
  - Windowed method `fir1, fir2`
  - Constrained Least Squares `fircls, fircls1`
  - Complex, nonlinear phase `cremez`
- IIR
  - Butterworth, Chebyshev Type I and II, Elliptic
    `butter, cheby1, cheby2, ellip`
  - Piecewise linear magnitude approx. `yulewalk`
  - Arbitrary Mag. & Phase `invfreqz`
  - Generalized Butterworth (lowpass only) `maxflat`

Other M-files in the Signal Proc. Toolbox

- Spectral Analysis - estimate Power Spectral Density
  - Welch's method (overlapped modified periodograms) `psd`
  - Maximum entropy method (AR modeling) `pmem`
  - MUSIC (eigenanalysis based) `pmusic`
  - Multitaper (discrete prolate spheroidal sequences) `pmtm`

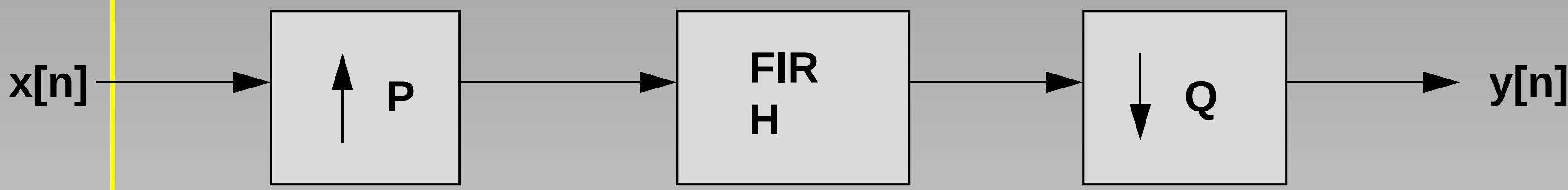
- Parametric Modeling - find AR or ARMA model for signal
  - AR model via autocorrelation technique `lpc`
  - ARMA `prony`
  - iterative ARMA `stmcb`

and More M-files in the Signal Proc. Toolbox

- Graphic User Interface (GUI) `sptool`

- Other
  - Correlation functions `xcorr, xcov`
  - Hilbert transform `hilbert`
  - Spectrogram `specgram`
  - Resampling `resample, upfirdn`
  - Alternate Filtering schemes `filtfilt, fftfilt`

upfirdn - Efficient Multirate FIR Filter bank Implementation



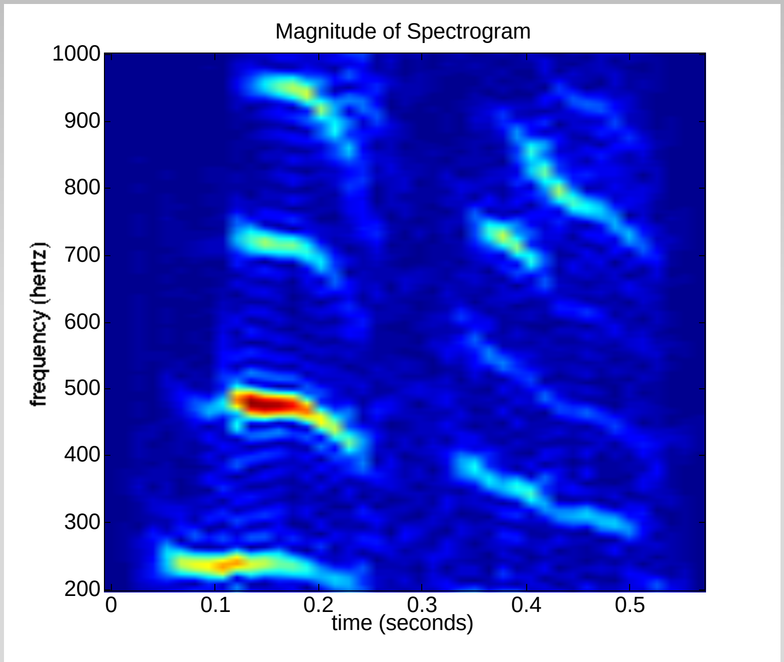
- Syntax:

```
y = upfirdn(x,h,p,q)
```

- x and h can be arrays to implement BANKS of filters

upfirdn Example: Compute spectrogram over range of frequencies

```
load mtlb
f = (200:5:1000);  % frequencies in Hz
w = hamming(256);
h = exp(-j*(0:255)'*f*2*pi/Fs);  % DFT filter bank
s = upfirdn(mtlb,h,1,100);  % compute DFT every 100 samples
t = (0:100:length(mtlb)+256)/Fs;
imagesc(t,f,abs(s.')), axis xy
```



New in MATLAB 5

- Multidimensional Arrays
  - fftn - N-dimensional FFT
  - convn - N-dimensional convolution (filtering)
- Non-double data containers - very useful for image and video work
  - uint8, int8
- Filter works on array inputs - filters each column

Other Signal Processing Relevant Toolboxes

- Image Processing `images`
- Higher Order Spectral Analysis (HOSA) `hosa`
- Wavelets `wavelet`
- Statistics `stats`
- System Identification `ident`
- Communications `comm`
- Optimization `optim`
- Symbolic Math `symbolic`
- Control Systems `control`
- Neural Networks `nnet`
- Fuzzy Logic `fuzzy`

For a list of available functions, type `help signal`, `help wavelet`, etc.

Some Symbolic Basics in MATLAB 5

- Define some symbolic variables using ‘syms’ command

```
» syms x a T w
» int(exp(a*x),0,T)
ans =
1/a*exp(T*a)-1/a
» solve(x^3+a)
ans =
[                              (-a)^(1/3)]
[ -1/2*(-a)^(1/3)-1/2*i*3^(1/2)*(-a)^(1/3)]
[ -1/2*(-a)^(1/3)+1/2*i*3^(1/2)*(-a)^(1/3)]
» int(exp(sqrt(-1)*w*x),-T/2,T/2)
ans =
-i/a*exp(1/2*i*T*w)+i/a*exp(-1/2*i*T*w)
» y = simplify(ans)
y =
2*sin(1/2*T*w)/w
»limit(y,w,0)
ans =
T
```

INTEGRATE

SOLVE EQUATION(S)

FOURIER TRANSFORM OF SQUARE PULSE

SIMPLIFY EXPRESSIONS

TAKE LIMITS