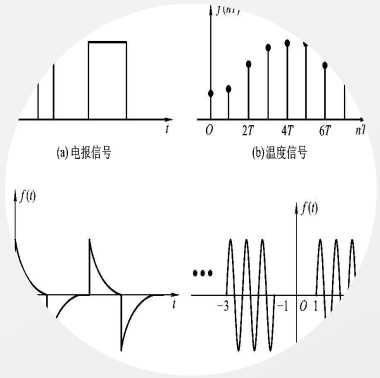
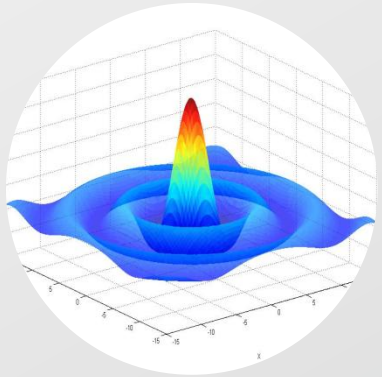
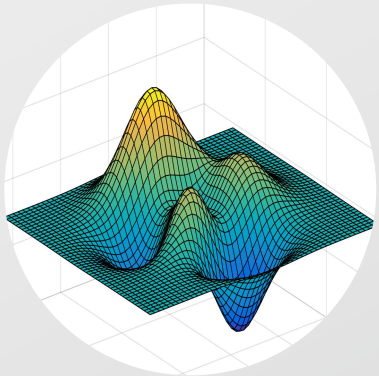
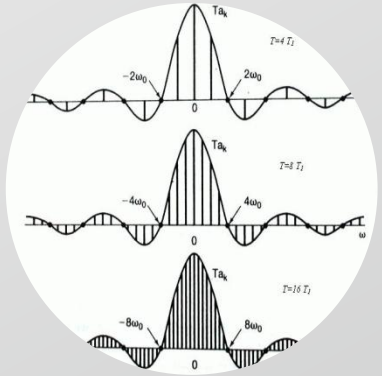






# Lab5 Preparation for Project 1（Filter）
# 项目1的准备工作（滤波器）

主讲老师：王小静
办公地点：一教131

# Time Domian *VS* Frequency Domain

## ➢ Time Domian

$x(t) \Rightarrow \boxed{h(t)} \Rightarrow y(t)$

$$x(t) * h(t) = y(t)$$

*Convolution*

*UnitImpulse Response*

VS

## ➢ Frequency Domain

$X(j\omega) \Rightarrow \boxed{H(j\omega)} \Rightarrow Y(j\omega)$

$$X(j\omega) \cdot H(j\omega) = Y(j\omega)$$

*Multipilication*

*Frequency Response*

$$X(j\omega) = \int_{-\infty}^{+\infty} x(t) e^{-j\omega t} dt$$

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} X(j\omega) e^{j\omega t} d\omega$$

## ➢ Time Domian

$x(t) \Rightarrow \boxed{h(t)} \Rightarrow y(t)$

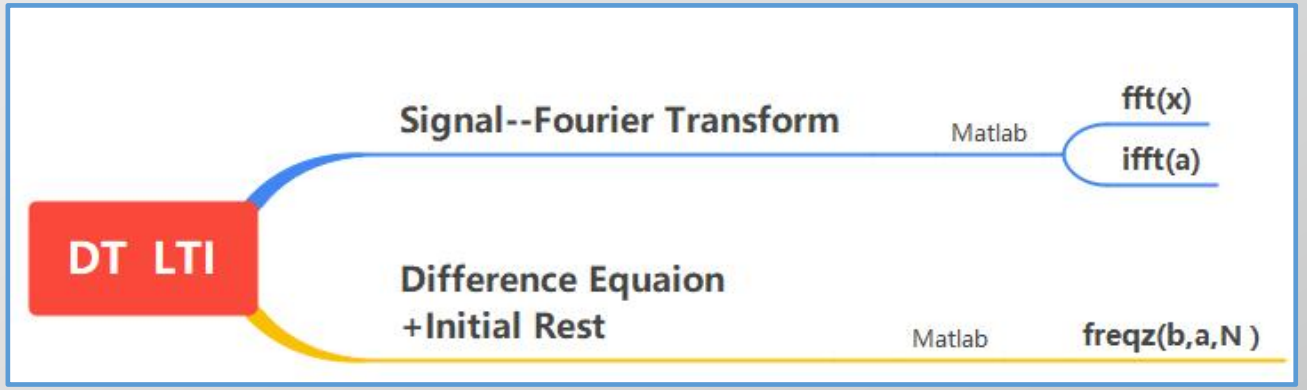
$x(t) * h(t) = y(t)$

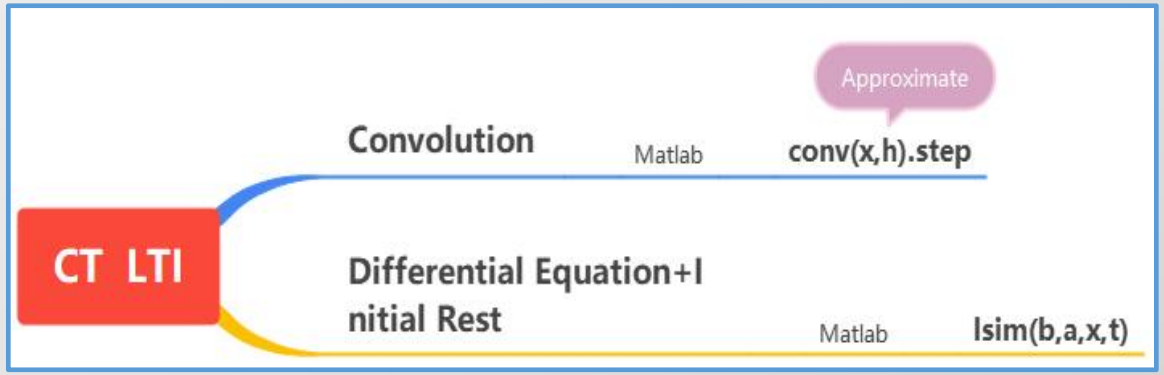
## ➢ Frequency Domain

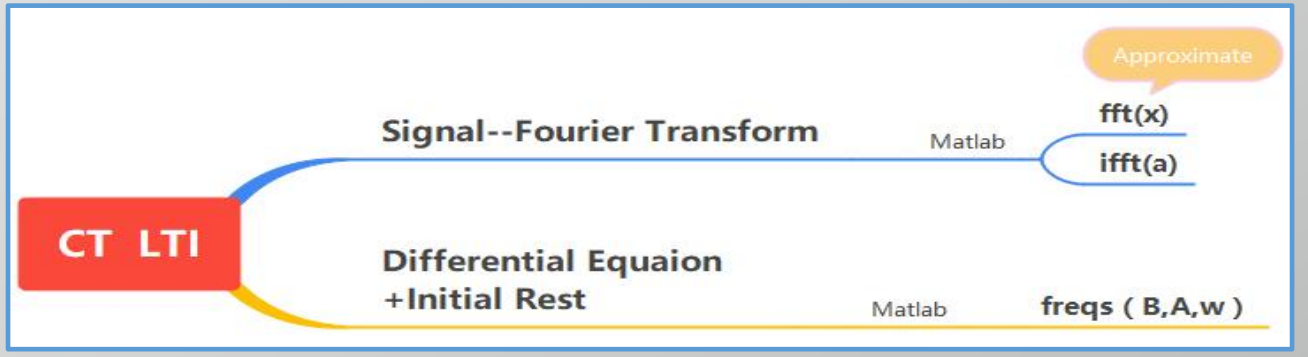
$X(j\omega) \Rightarrow \boxed{H(j\omega)} \Rightarrow Y(j\omega)$

$X(j\omega) \cdot H(j\omega) = Y(j\omega)$

VS

# Contents

Catalogue

1 Design Butterworth filters via Matlab

2 Use FIR filters to

- generate speech-shaped noise
- extract signal envelope

3 Adjust the signal intensity

- adjusting signal-to-noise ratio (SNR) level
- normalizing signal energy

# Part1 ：Butterworth Filters Design

- The Butterworth filter is a type of signal processing filter designed to have as flat a frequency response as possible in the passband. It is also referred to as a maximally flat magnitude filter.
- The frequency response of the Butterworth filter is maximally flat (i.e. has no ripples) in the passband and rolls off towards zero in the stopband.
- The Butterworth filter rolls off more slowly around

the cutoff frequency than the Chebyshev filter

or the Elliptic filter, but without ripple.

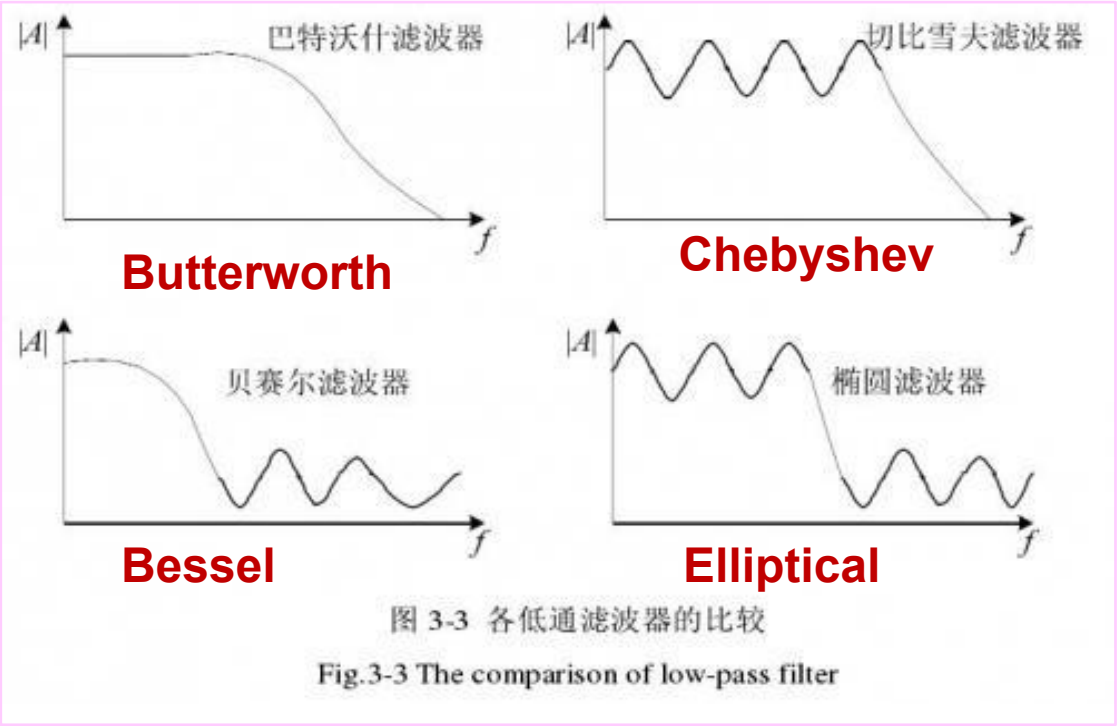
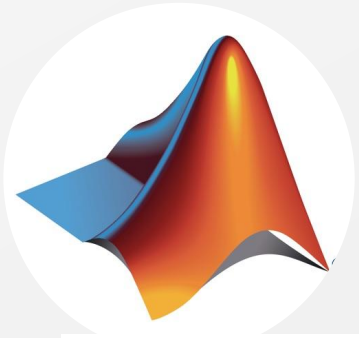



图 3-3 各低通滤波器的比较

Fig.3-3 The comparison of low-pass filter

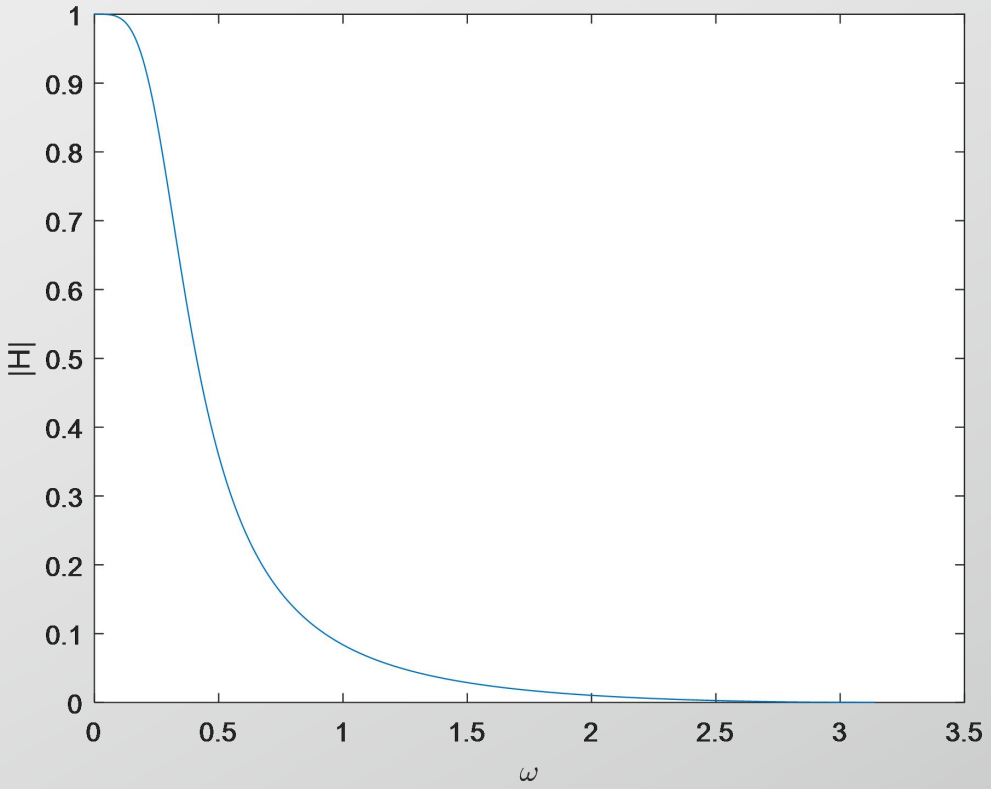


# Design filters via Matlab function butter

- **[b,a]=butter(n,Wn)**
  - Design an order n low-pass digital Butterworth filter with ***normalized cutoff frequency*** Wn. Wn must be 0.0 < Wn < 1.0, with 1.0 corresponding to half the sample rate.
  - Return the filter coefficients in length n+1 vectors b and a.
- **[b,a]=butter(n,Wn,'ftype')**
  - Design a highpass, lowpass, or bandstop filter, where the string ‘ftype’ is ‘high’, ‘low’, ‘bandpass‘ or ‘stop’。

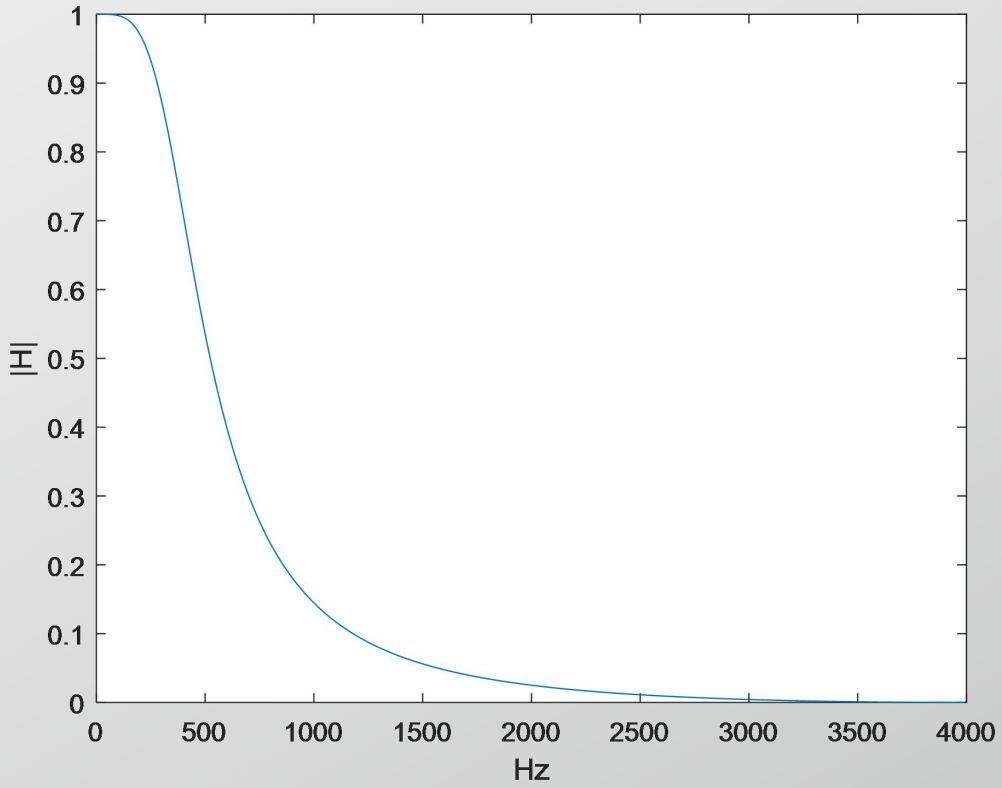
# Example:

- fs=8000;
- %design filter
- [b, a]=butter(2,400/(fs/2));
- %frequency response of the filter



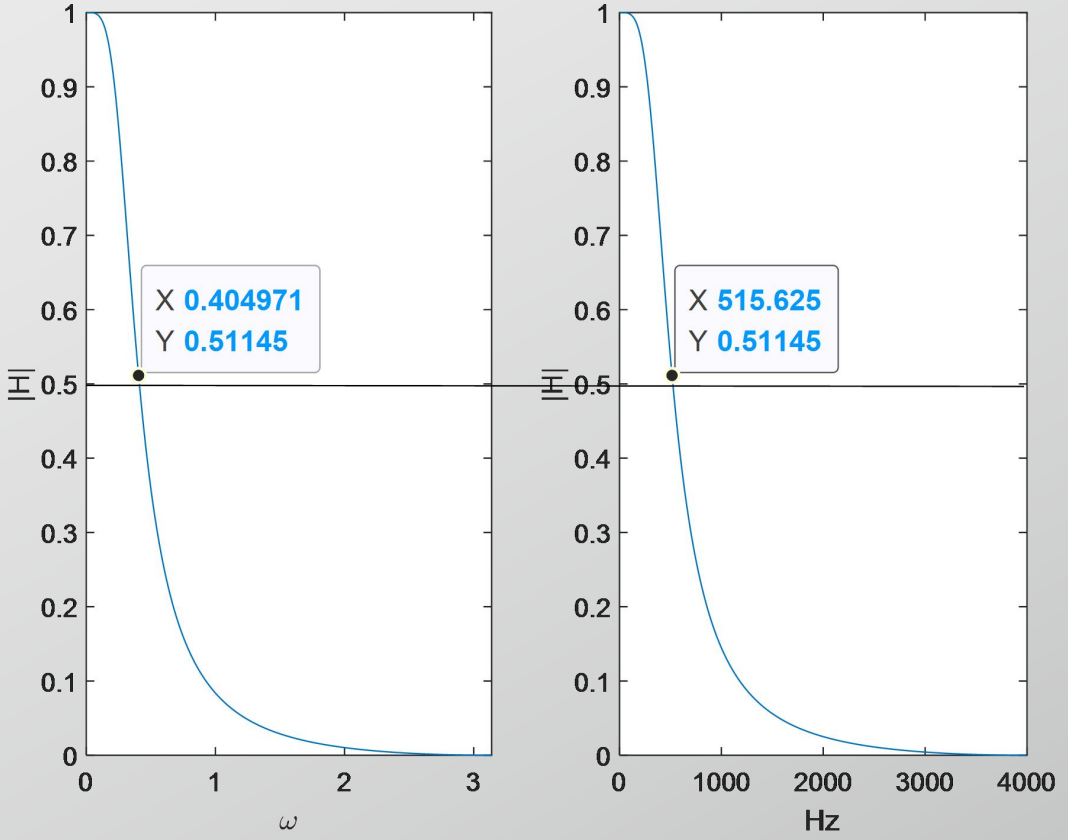
# Example:

- fs=8000;
- %design filter
- [b, a]=butter(2,400/(fs/2));
- %frequency respons of the filter
- **[h,f]=freqz(b,a,512,fs)**;
- plot(f,abs(h)); % in dB scale
- axis([0 5000 -50 5]);
- xlabel('Frequency (Hz) ');
- ylabel('Magnitude');



# Digital frequency and Physical frequency

- fs=8000;
- %design filter
- [b, a]=butter(2,400/(fs/2));
- %frequency response of the filter

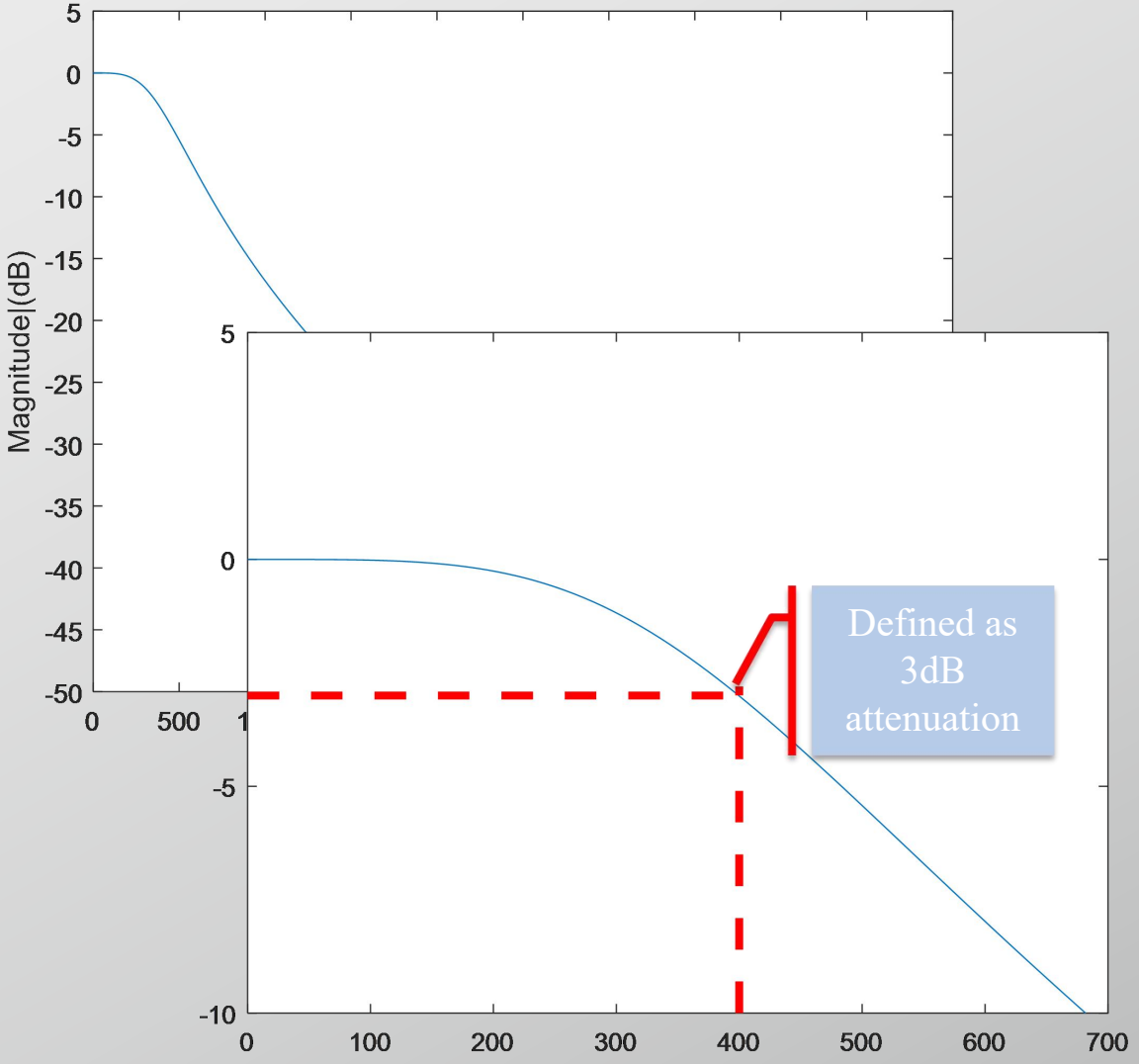



$$\pi \leftrightarrow f_s/2$$

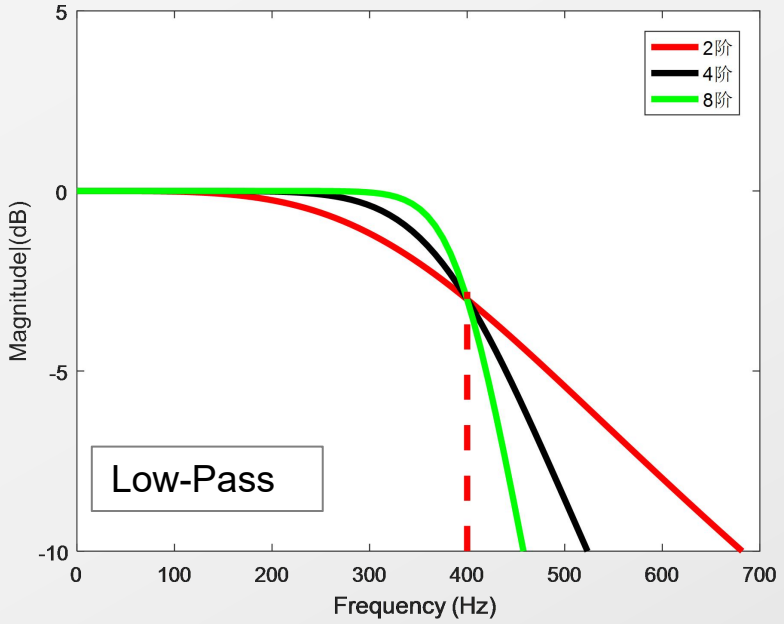
# Example:

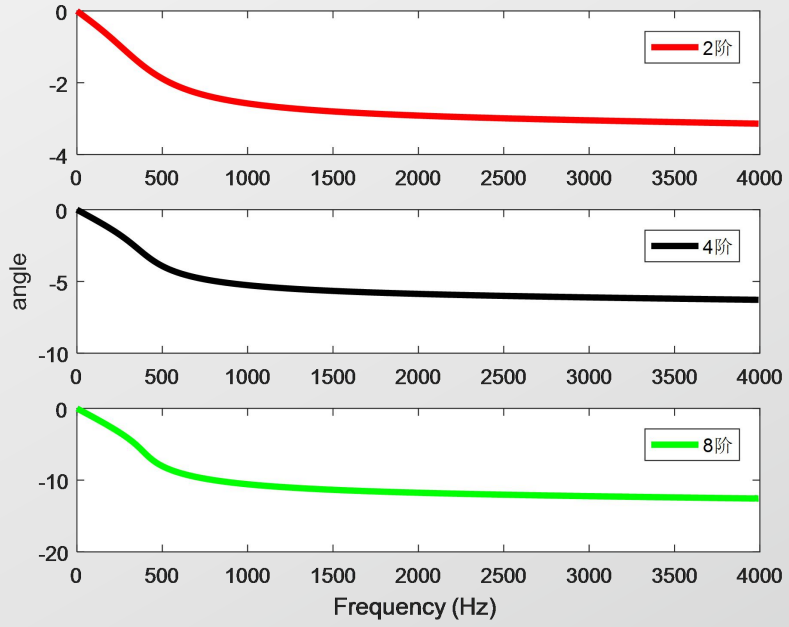
- fs=8000;
- %design filter
- [b, a]=butter(2,400/(fs/2));
- %frequency respons of the filter
- **[h,f]=freqz(b,a,512,fs)**;
- plot(f,20*log10(abs(h))); % in dB scale
- axis([0 5000 -50 5]);
- xlabel('Frequency (Hz) ');
- ylabel('Magnitude');

How about ‘high-pass’ and ‘band-pass’?
How about high-order n?

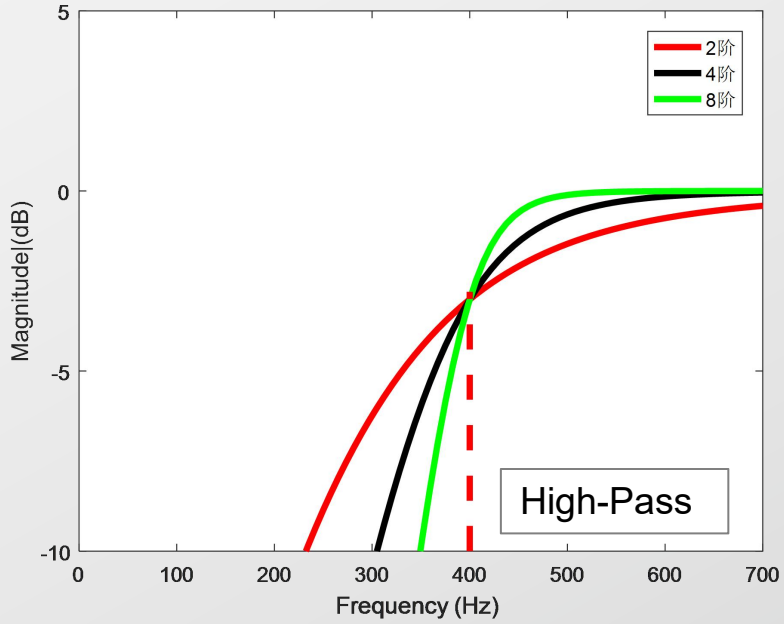


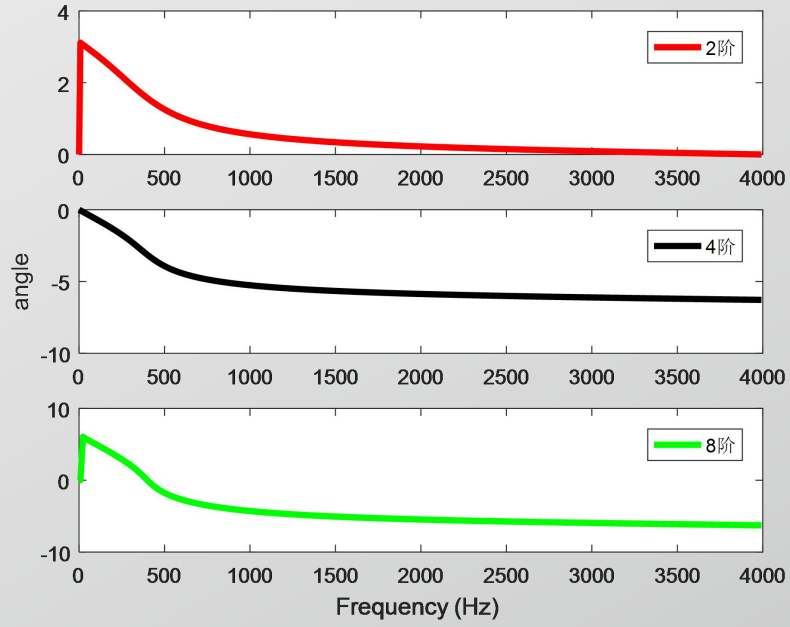
[b, a]=butter(n,400/(fs/2));



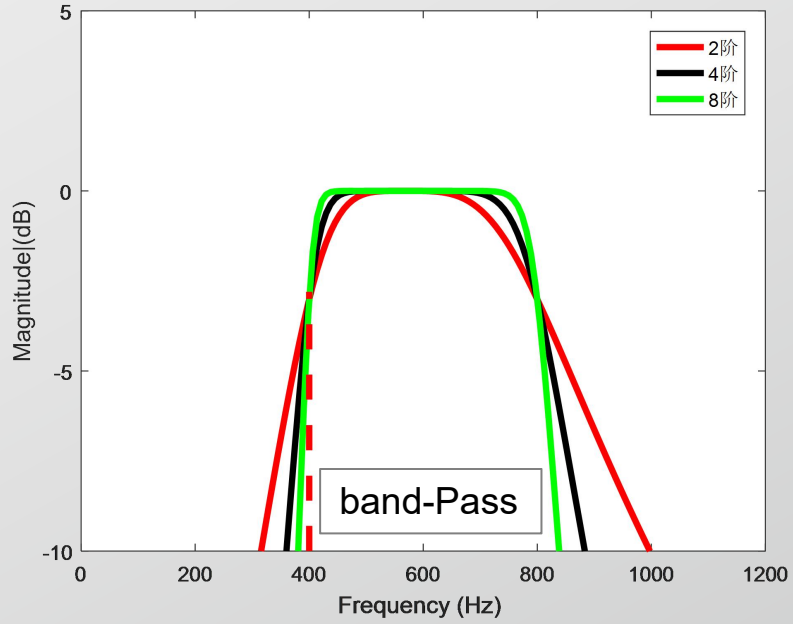



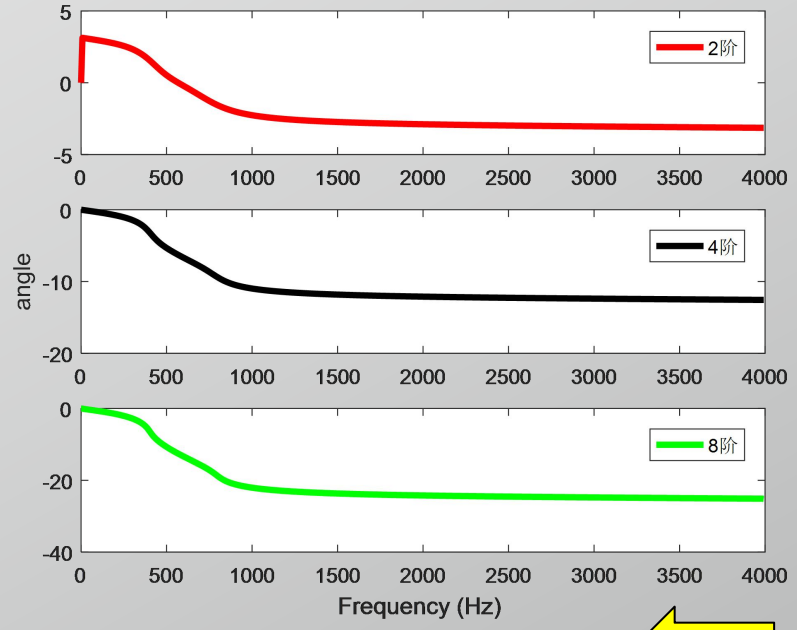

[b, a]=butter(n,400/(fs/2),'high');




[b, a]=butter(n,[400,800]/(fs/2),'bandpass');



# About the coordinate system

## Advantages of logarithmic coordinate systems

Frequency domain fro CT signals - the module is represented by decibels, 20log, and the horizontal axis is the logarithmic frequency coordinates.

1: The multiplication is changed into addition, and it is unified with the phase calculation method;

2: Details can be displayed in a wide dynamic range;

3: The shape of a specific response curve will not change according to the weight of frequency;

4: For a CT LTI system described by differential equations, the approximate map of logarithmic frequencies by logarithmic modules is often easily drawn by using asymptotic lines.

Frequency domain for DT signals - the module is also represented by decibel, 20log, but the horizontal axis does not use logarithmic frequency coordinates.

1: Limited frequency range, $2\pi$ symmetry;

2: The linear asymptotic method does not hold;

In the actual system, linear and logarithmic coordinate representation can be selected according to the convenience of use.

# The Sampling Theorem

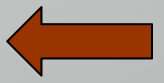
*Let $x(t)$ be a band-limited signal with*

$$X(j\omega) = 0 \text{ for } |\omega| > \omega_M.$$

*Then, $x(t)$ is uniquely determined by its samples $x(nT)$ or $x_p(t)$ if*

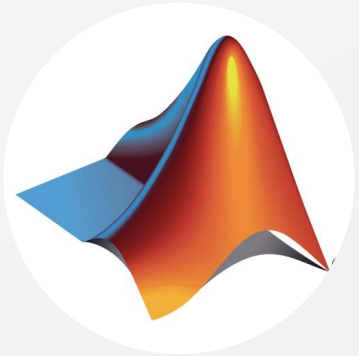
$$\omega_s = \frac{2\pi}{T} > 2\omega_M,$$

*where $2\omega_M$ is referred to as the Nyquist rate.*



# Part2 : FIR Filters Design

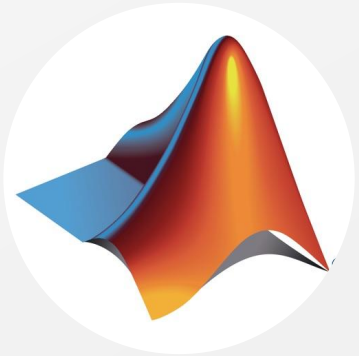
- In signal processing, a finite impulse response (FIR) filter is a filter whose impulse response (or response to any finite length input) is of finite duration, because it settles to zero in finite time. This is in contrast to infinite impulse response (IIR) filters, which may have internal feedback and may continue to respond indefinitely (usually decaying).
- FIR filters are inherently stable, and can easily be designed to be linear phase by making the coefficient sequence symmetric.
- **fir1** : design lowpass, bandpass, or multiband FIR filter .
- **fir2** : design window FIR filters with arbitrary frequency response.



# Design filters via fir1

- **b = fir1(n,Wn)**
  - uses a Hamming window to design an nth-order lowpass, bandpass, or multiband FIR filter with linear phase. The filter type depends on the number of elements of Wn.

- **b = fir1(n,Wn,ftype)**
  - designs a lowpass, highpass, bandpass, bandstop, or multiband filter, depending on the value of ftype and the number of elements of Wn.

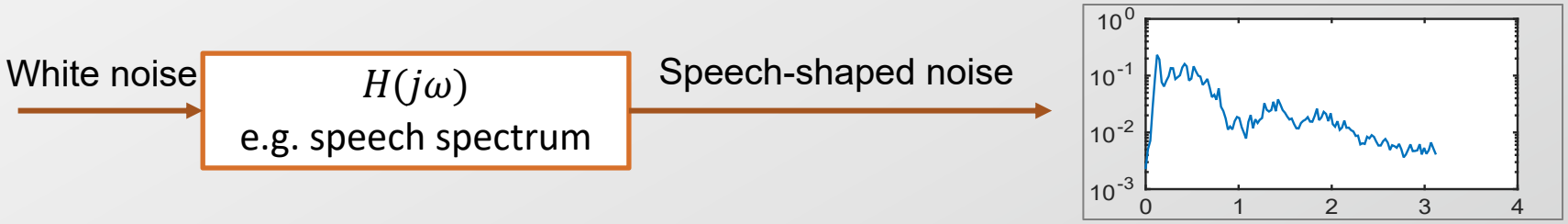


# Design filters via fir2

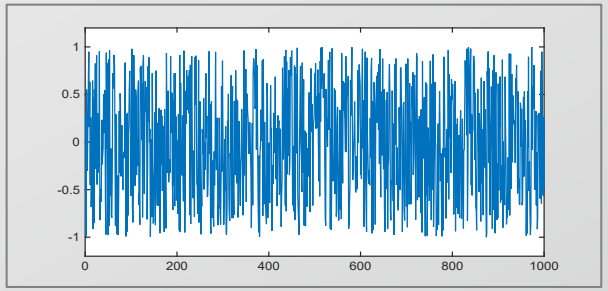
- **b = fir2(n,f,m)**
  - returns an nth-order FIR filter with frequency-magnitude characteristics specified in the vectors f and m.
  - The amplitude frequency characteristics of this filter are determined by f and m veators, where **f is the normalized frequency vector** and **m is the amplitude at the corresponding frequency point**. f and m have the same length, and the first and last components of f are 0 and 1;

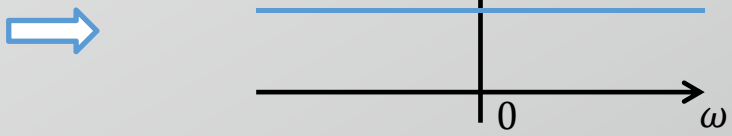
# Application1 ：Generate speech-shaped noise by FIR filter

- Speech-Shaped Noise (SSN), can be obtained by white noise processed by spectral filter.




- White noise






```
N=1000; noise = 1-2*rand(1,N);
```

# STEP 1

## (A) generate long-term spectrum of speech signal

✓concatenate 10 speech waveforms into one signal

```
sig =[x,x,x,x,x,x,x,x,x,x];  % x is a row vector, what if x is a columnvector?
```

Or, **repmat**(x,1,10)

✓estimate the power spectral density of the speech signal

[Pxx,w] =**pwelch**(sig, window,noverlap,NFFT, fs) : returns the power spectral density (PSD) estimate, pxx, of the input signal,

✓plot the power spectral density

Matlab谱分析的pwelch方法

```
plot(w,Pxx)
```

# STEP 2

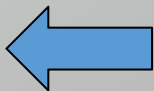
## (B) generate filter coefficients

```
% fir2: generate coefficients b, a=1,
b = fir2(3000,w/(fs/2),sqrt(Pxx/max(Pxx)));
%check frequency response here
[h,wh] = freqz(b,1,128);
```

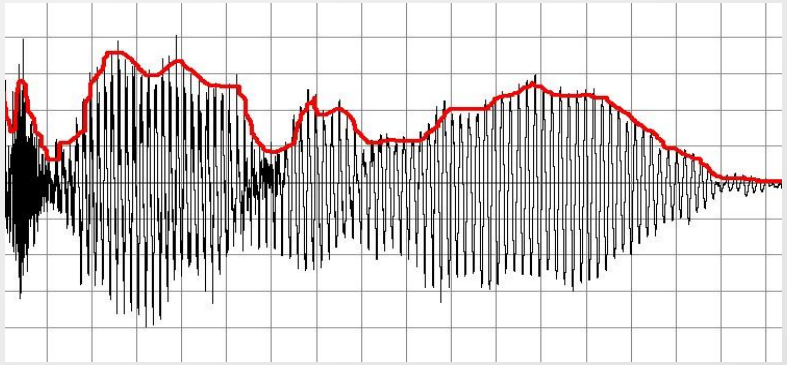
## (C) perform filtering on white noise signal

```
y= filter(b,1,noise);
```

Now, we'll get the speech-shaped noise

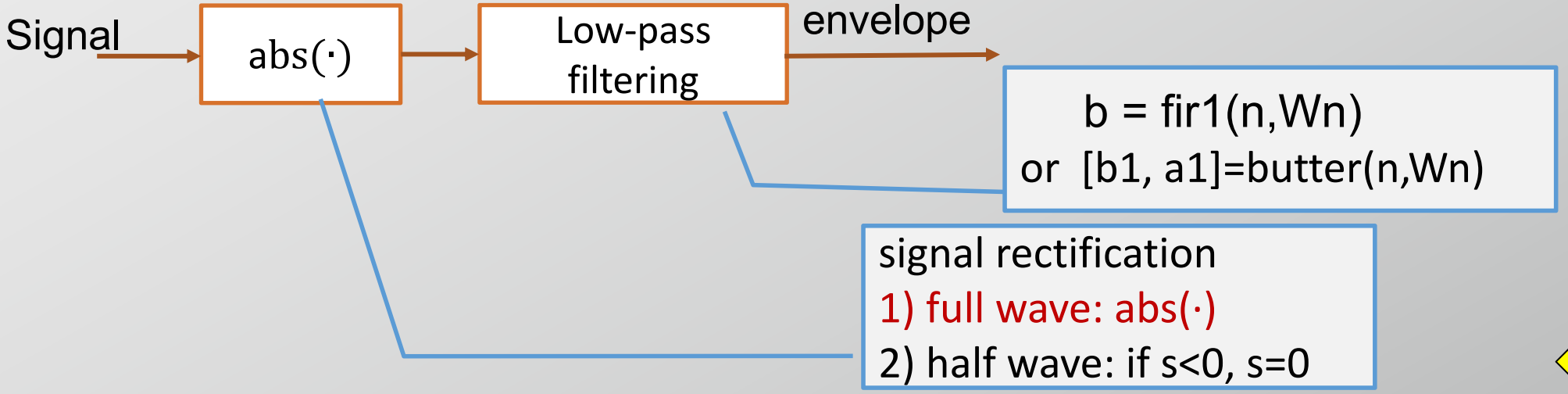


# Application2：Extract speech envelope (包络) waveform by FIR filter



Original waveform
Envelope

https://commons.wikimedia.org/wiki/File%3AC_Envelope_follower.jpg



# Examples:Extract speech envelope by lowpass filter

```
clear all; clc; close all;
t=0:0.01:10; % samples
f1=5;  %carrier
f2=0.1;  %signal
plot(t,cos(2*pi*f1*t));
plot(t,cos(2*pi*f2*t));
x=cos(2*pi*f1*t).*cos(2*pi*0.1*t);
%rectification
x1=x;
for n=1:length(t)
    if(x(n)<0)
    x1(n)=0;
    else
        x1(n)=x(n);
    end
end
x2=abs(x);
plot(t,x);title('x，原信号')
plot(t,x1);title('x1，if s<0, s=0')
plot(t,x2);title('x2，abs(x)')
```

Try to plot their spectrums

```
fs=100;
%fir1 to design lowpass filter,n=32
b2=fir1(32,1/(fs/2));
a2=1;
%butter to design lowpass filter,n=6
[b1, a1]=butter(6,2/(fs/2));
```

```
%Extract envelope by FIR filter
y_envelope=filter(b2,a2,x);
y_envelope=filter(b2,a2,x1);
y_envelope=filter(b2,a2,x2);
```

contrast

```
%Extract envelope by butter filter
y_envelope=filter(b1,a1,x);
y_envelope=filter(b1,a1,x1);
y_envelope=filter(b1,a1,x2);
```

plot the origial signal and their envelope  in the same figrue

# Part3：SNR & Adjust Signal Intensity ）

- **Signal-to-noise ratio (abbreviated SNR or S/N) is a measure used in science and engineering that compares the level of a desired signal to the level of background noise. It is defined as the ratio of signal power to the noise power, often expressed in decibels.。**
  - $SNR = 10log_{10}(E_{signal}/E_{noise}) = 20log_{10}(\sqrt{E_{signal}}/\sqrt{E_{noise}})$
  - *E* is energy, and *SNR* in dB scale
  - In Matlab: SNR=20*log10(norm(sig)/norm(noise)) % sig and noise must have the same length

n = norm(X) returns the 2-norm of input X and is equivalent to norm(X,2). If X is a vector, norm(X) is the Euclidean distance, just $\sqrt{E}$。

help norm ----- Matrix or vector norm.

向量和矩阵的各种范数比较

# Applicaion： Adjust Signal Intensity to Meet SNR Requirement

- Energy adjustment
  - **Energy normalization**
  - y'=y/norm(y)*norm(x);  % then y'energy is normalized to 1；
  - **Energy adjustment**
  - y'=y/norm(y)*norm(x); % then y has the same energy as x
  - y'=y/norm(y)*norm(x)*2; % then y has the twice energy of x

**How to adjust Signal or noise energy to meet SNR requirement when adding noise to a signal?**

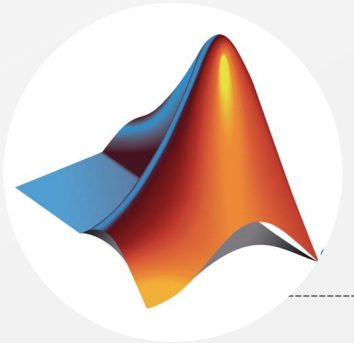
- 1:  Adjust the energy of noise first

$$SNR = 10log_{10}(E_{signal}/E_{noise}) = 20log_{10}(\sqrt{E_{signal}}/\sqrt{E_{noise}}) \Rightarrow E_{noise} = f(SNR, E_{signal})$$

$$norm() = \sqrt{E} \Rightarrow norm(signal) = \sqrt{E_{signal}},\ \ norm(noise) = \sqrt{E_{noise}}$$

- 2:  add the noise to signal

**Verify the resultant SNR after adjusting the noise energy**



# More on wave file …

- **Read wave file ——*.wav**
  - [y, fs] = audioread(filename) ,[y,fs] = audioread(filename,samples)
    - y: sampled data, signal, fs: sampling rate in Hertz,
  - [y,fs] = audioread(filename,samples)
    - reads the selected range of audio samples in the file, where samples is a vector of the form [start,finish].

- **Play wave file**
  - sound(y)
    - sends audio signal y to the speaker at the default sample rate of 8192 hertz.
  - sound(y, fs)
    - sends audio signal y to the speaker at sample rate Fs.
  - player = audioplayer(y, fs)；play(player)
    - audioplayer：Create object for playing audio，
    - paly：Play audio from audioplayer object

- **Save wave file**
  - audiowrite(filename,y,fs)
    - writes a matrix of audio data, y, with sample rate Fs to a file called filename. The filename input also specifies the output file format. The output data type depends on the output file format and the data type of the audio data, y.
    - eg:

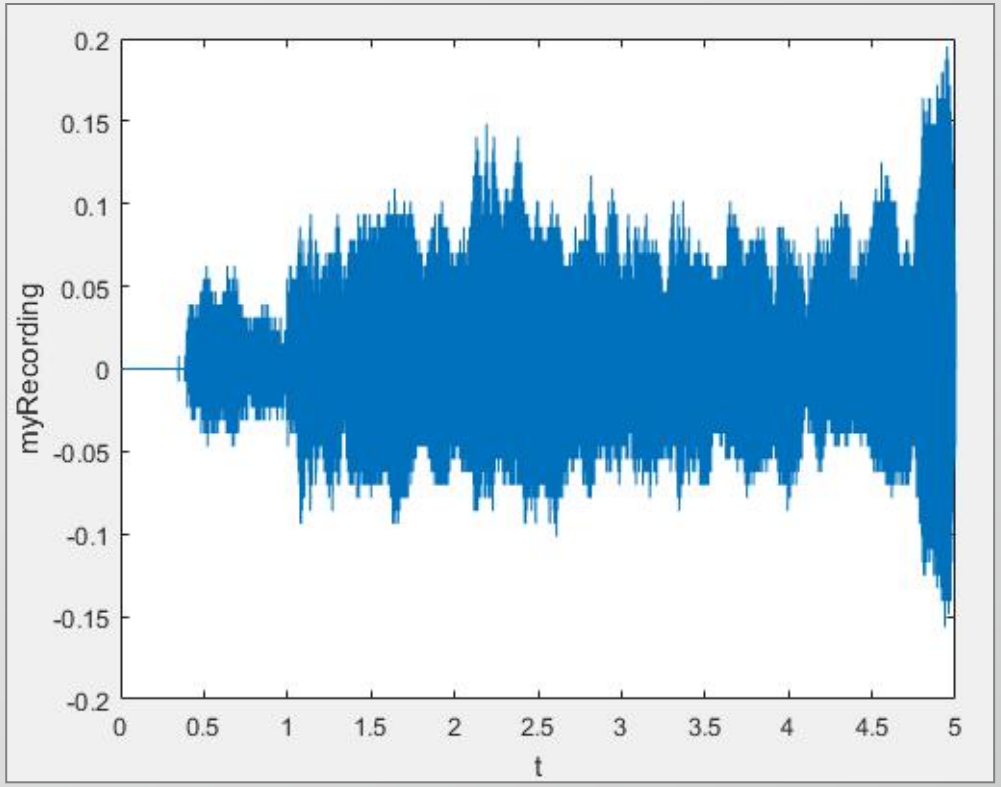
```
load handel.mat
filename = 'handel.wav';
audiowrite(filename,y,Fs);
```

```
load handel.mat
filename = 'handel.flac';
audiowrite(filename,y,Fs);
```

- **Record sound using PC-based audio input device**
  - recorder = audiorecorder
    - creates an 8000 Hz, 8-bit, 1-channel audiorecorder object.
  - recorder = audiorecorder(Fs,nBits,nChannels)
    - sets the sample rate Fs (in Hz), the sample size nBits, and the number of channels nChannels.
  - recordblocking(recorderObj, length)
    - Record audio to audiorecorder object, holding control until recording completes

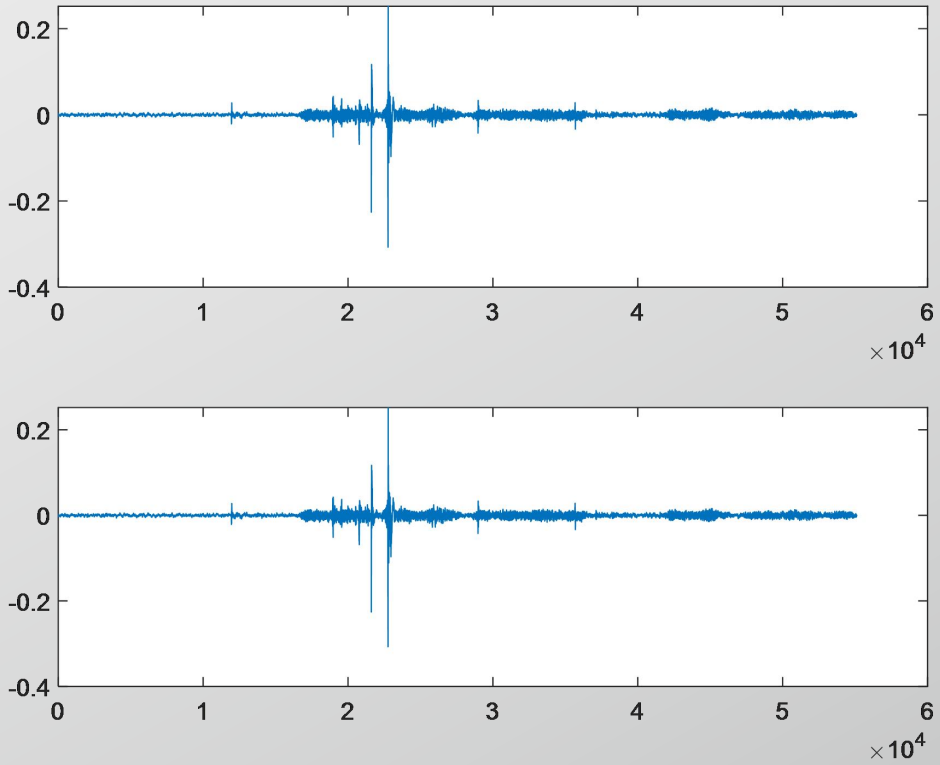
# An example

```
% Record your voice for 5 seconds.
recObj = audiorecorder(11025,8,1);
disp('Start speaking.')
recordblocking(recObj,5);
disp('End of Recording.');
% Play back the recording.
play(recObj);
% Store data in double-precision array
myRecording = getaudiodata(recObj);
plot(myRecording); % Plot the waveform.
```




See Matlab documentation for more details

```
clc,clear,close all
fs = 11025; nbits = 16; length = 5;
% Record your voice for 5 seconds.
recObj = audiorecorder(fs,nbits,1);
disp('Start speaking.')
recordblocking(recObj, length);
disp('End of Recording.');
% Play back the recording.
play(recObj);
% Store data in double-precision array
myRecording = getaudiodata(recObj);
subplot(2,1,1); plot(myRecording); % Plot the waveform.
audiowrite('Myvoice.wav', myRecording, fs)
[sigread,fsread] = audioread('Myvoice.wav');
subplot(2,1,2); plot(sigread); % Plot the waveform.
```



# Lab5 Assignments

1 Generate a speech-shaped noise (SSN), and plot the spectrum of the speech signal and SSN .

tips:

1: two audio files in BB: 'C_01_01.wav' & 'C_01_02.wav'

2: power spectral density estimation: [Pxx,w] = pwelch(sig,Windows,noverlap,nfft,fs);

let nfft=512; noverlap = nfft/2;

Windows = hamming(nfft); or Windows = hann(nfft);

Matalb help

**2**

Read a speech signal x(t), adjust the SNR (x(t) to the above SSN) to −5 dB, let $y = x + \text{SSN}$, and normalize the energy of y equals to the energy of x. plot y.

# 3

Extract speech envelope

1) with 2nd–order lowpass Butterworth filter and cutoff frequency fcut= 100, 200, and 300 Hz. Plot these three envelope waveforms in one plot, and describe the difference among them.

2)with 2nd and 6th–order lowpass Butterworth filter and cutoff frequency 200 Hz. Plot these two envelope waveforms in one plot, and describe the difference between them.

# 4 Record an audio and save it as a wave file

# Lab Schedule

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 第10周 春季学期 | 21 廿四 | 22 廿五 | 23 廿六 | 24 廿七 | 25 廿八 | 26 廿九 | 27 三十 |
| 5月 | 第11周 春季学期 | 28 四月 | 29 初二 | 30 初三 | 1 劳动节 | 2 初五 | 3 初六 | 4 青年节 |
| | 第12周 春季学期 | 5 立夏 | 6 初九 | 7 初十 | 8 十一 | 9 十二 | 10 十三 | 11 母亲节 |
| | 第13周 春季学期 | 12 十五 | 13 十六 | 14 十七 | 15 十八 | 16 十九 | 17 二十 | 18 廿一 |
| | 第14周 春季学期 | 19 廿二 | 20 廿三 | 21 小满 | 22 廿五 | 23 廿六 | 24 廿七 | 25 廿八 |
| | 第15周 春季学期 | 26 廿九 | 27 五月 | 28 初二 | 29 初三 | 30 初四 | 31 端午节 | 1 儿童节 |
| | 第16周 春季学期 | 2 初七 | 3 初八 | 4 初九 | 5 芒种 | 6 十一 | 7 十二 | 8 十三 |

← lab 5

← Intro to project 1

← Intro to project 2

← Q&A

← Presentation 1

← Presentation 2

**Coding Test**

**3/4 members a group**