7. Signal processing

- Using audio data
- Fourier transform
- Noise reduction

Using audio data

- `audioread` reads audio signals recorded in a file
- `sound` plays audio signals

```
>> [f,freq]=audioread('test.wav');
>> size(f)
ans =
   211289           1
>> freq
freq =  44100
>> x1=length(f)/freq
ans =   4.7911
>> x=linspace(0,x1,length(f));
>> plot(x,f);
>> sound(f,freq);
>> sound(f,0.7*freq);
```

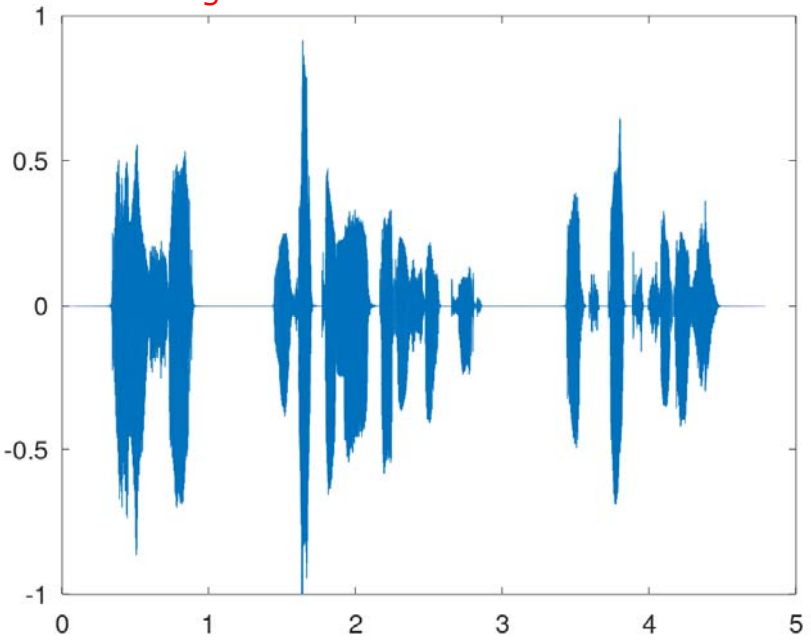
211 thousand sample points in 1 channel

sampling rate = 44100Hz (i.e., 44.1 thousand points per sec.)

length of the audio signal is about 4.79 sec.

play the audio signal

play at a different rate



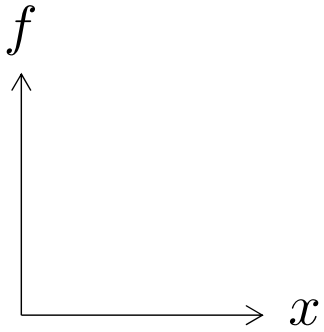
※ An example of a voice signal

Fourier transform (1/2)

- Fourier series expansion: A periodic signal can be represented as a linear *weighted* sum of sinusoidal waves of different frequencies

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} (a_k \cos kx + b_k \sin kx)$$

- Each weight in the linear sum is regarded as the component of the frequency associated with the weight in the original signal



https://commons.wikimedia.org/wiki/File:Fourier_transform_time_and_frequency_domains_(small).gif

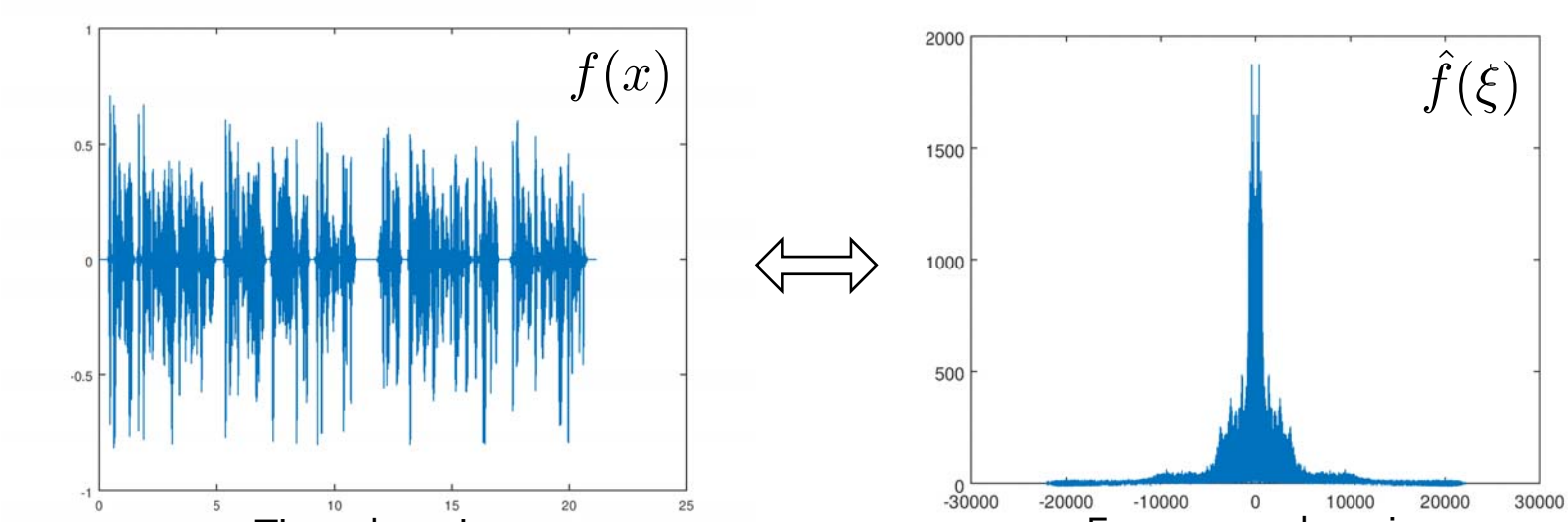
Fourier transform (2/2)

- A (periodic) signal can be represented by frequency components
- Generalizing this idea, Fourier transform represents a signal as distribution of frequency components

$$\hat{f}(\xi) := \int_{-\infty}^{\infty} f(x)\, e^{-2\pi i x \xi}\, dx$$

- The representation $\hat{f}(\xi)$ in the frequency domain can be transformed back to that $f(x)$ in the temporal domain

$$f(x) = \int_{-\infty}^{\infty} \hat{f}(\xi) e^{2\pi i x \xi} d\xi$$



Computing frequency components

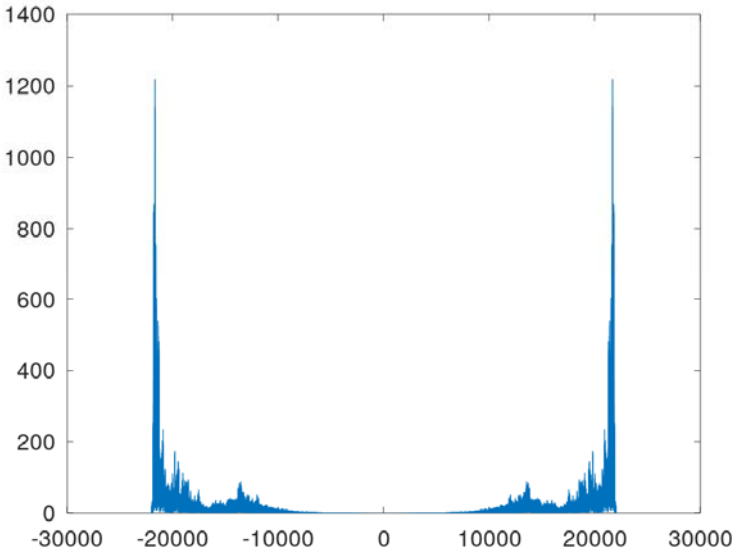
- `fft` performs Fourier transform (algorithm: *fast Fourier transform*) for an input signal

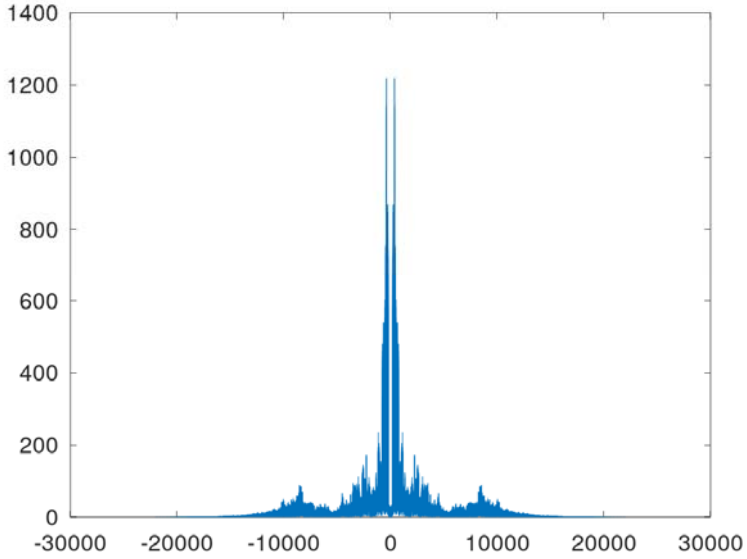
```
>> F=fft(f(:,1));
>> fs=size(F)
ans =
   211289        1
>> F(50000)
ans =  -5.5782 + 3.2176i
>> df=freq/length(F);
>> xi=-freq/2:df:freq/2-df;
>> plot(xi,abs(F))
```

※ `freq/2` in the specified range reflects the fact that frequency components higher than ½ of the sampling frequency (known as Nyquist frequency) cannot be correctly sampled;

```
>> Fshift=fftshift(F);
>> plot(xi,abs(Fshift))
```

※ `F` is a vector, in which the zero-frequency component is at the both ends, not at the center; `fftshift` shift F so that the zero-freq. component is at the center

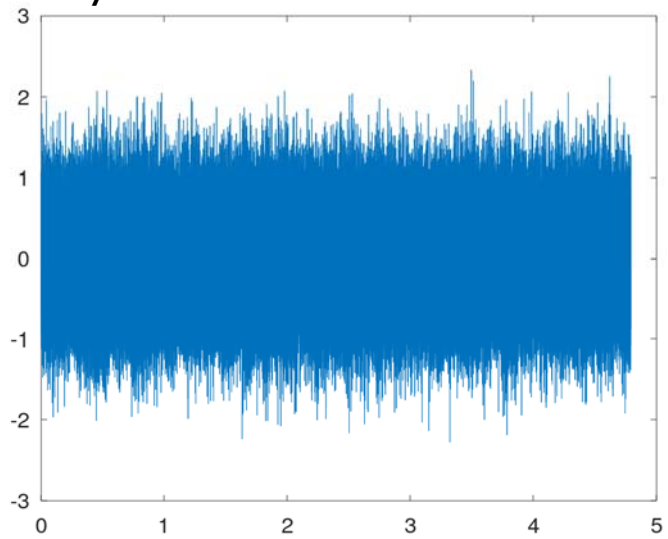




Noise reduction (1/2)

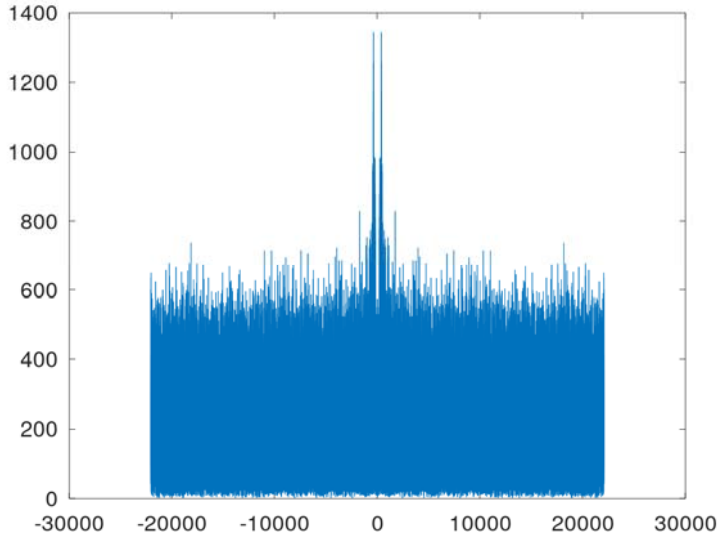
- Let's adding noises to a signal intentionally

```
>> f2=f+0.5*randn(size(f));
>> sound(f2,freq);
>> plot(x,f2);
```



- When seeing the signal in the frequency domain

```
>> F2shift=fftshift(fft(f2));
>> plot(xi,abs(F2shift))
```



Noise reduction (2/2)

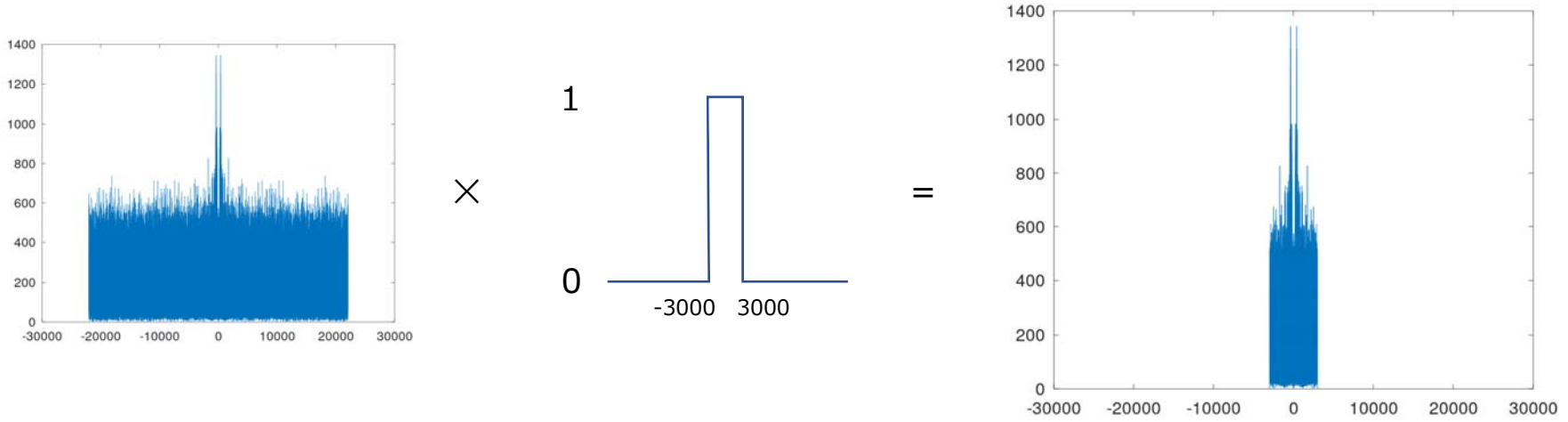
- Let's remove high-frequency components and perform inverse Fourier transform
  - Creating a filter eliminating frequency components lower than 3kHz

```
>> filter=abs(xi) < 3000;
```

※ See how results will change if you change the value of 3kHz

  - Element-wise multiplication between the signal and the filter, followed by application of inverse shift (`ifftshift`) and inverse transform (`ifft`)

```
>> f2filtered = ifft(ifftshift(F2shift.*filter'));
>> plot(xi,abs(F2shift.*filter')(:,1))
>> sound(f2filtered,freq)
```



Exercise 7.1

1. Create digital data of sinusoidal waves of different frequencies, play them as audio signals, and plot them in the time and frequency domain
2. Add one of the created sinusoidal waves to the voice signal (test.wav) and plot the resulting signal in the time and frequency domain
3. Consider a method for eliminating the added sinusoidal wave from the signal of Q2 as much as possible, and show the result by plots of the signals in the time and frequency domain