7. Signal processing　信号処理

- Using audio data　音声データの使い方
- Fourier transform　フーリエ変換
- Noise reduction　ノイズ低減処理

Using audio data　音声データの使い方

- `audioread` reads audio signals recorded in a file
  `audioread`関数で音声ファイルを読み込む
- `sound` plays audio signals
  `sound`関数で音声信号を再生する

```
>> [f,freq]=audioread('test.wav');
>> size(f)
ans =
   932591           2
>> freq
freq =  44100
>> x1=length(f)/freq
ans =   21.147
>> x=linspace(0,x1,length(f));
>> plot(x,f);
>> sound(f,freq);
>> sound(f,0.7*freq);
```

930 thousand sample points in 2 (left & right) channels
93万サンプリング点が左右の2チャンネル分

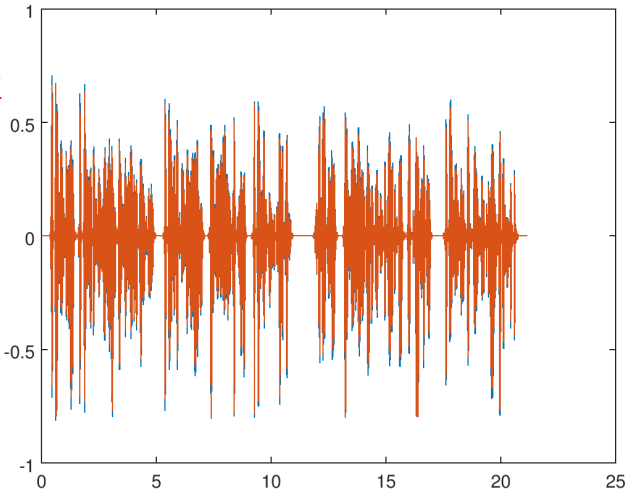
sampling rate = 44100Hz (i.e., 44.1 thousand points per sec.)
サンプリングレート= 44100Hz（すなわち、毎秒44100点）

length of the audio signal is about 21 sec.　音声信号の長さは約21秒

play the audio signal　音声信号の再生

play at a different rate
異なったレートでの再生

linspace (base, limit, n)
baseとlimitの間を，等間隔でn個に区切った要素をもつ 行ベクトルを返す.



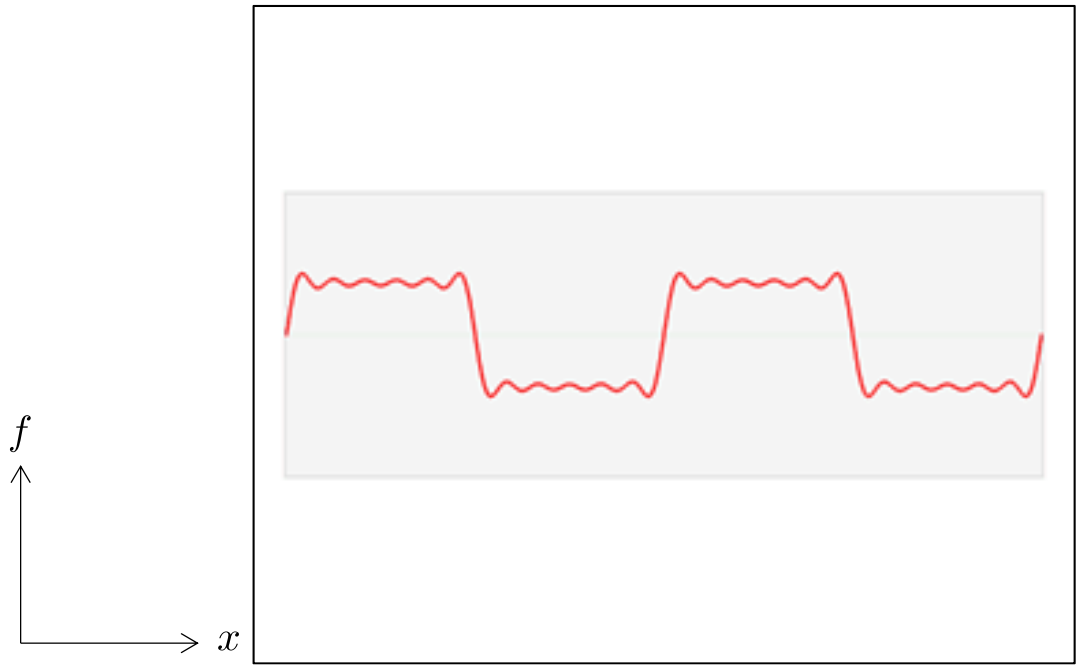
※ An example of a voice signal

Fourier transform (1/2)　フーリエ変換　(1/2)

- Fourier series expansion: A periodic signal can be represented as a linear *weighted* sum of sinusoidal waves of different frequencies
  フーリエ級数展開：周期信号は，異なる周波数の正弦波の線形加重和として表すことができる.

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

- Each weight in the linear sum is regarded as the component of the frequency associated with the weight in the original signal
  線形和の各重みは，元の信号の重みに関連する周波数の成分と見なされる.



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

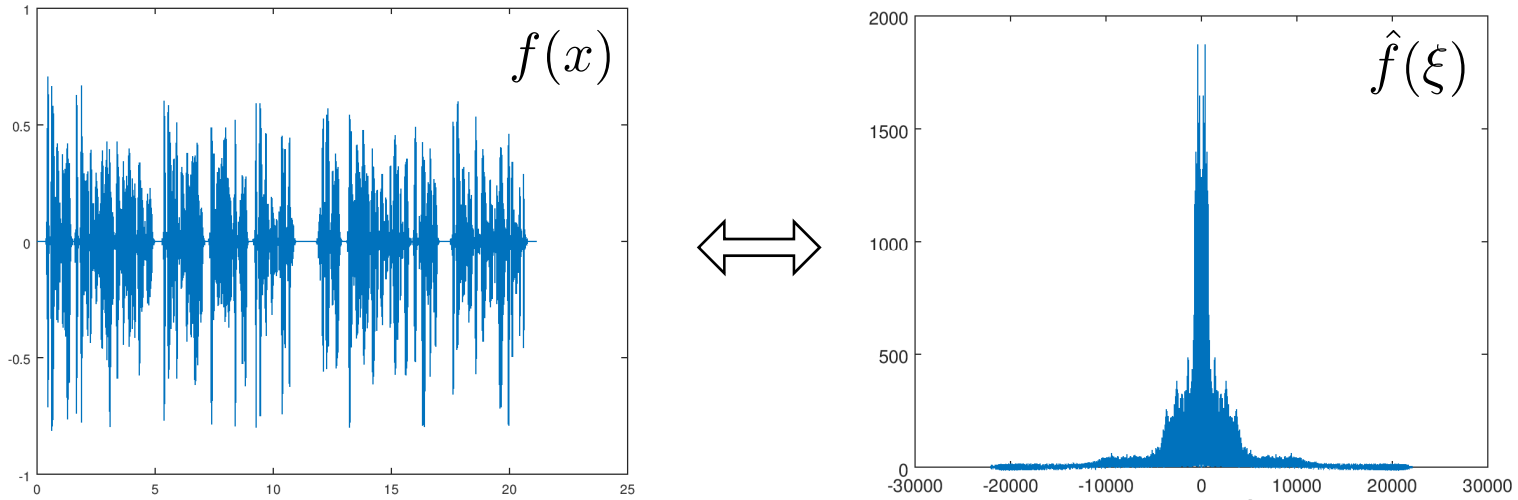
Fourier transform (2/2)　フーリエ変換　(2/2)

- 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

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



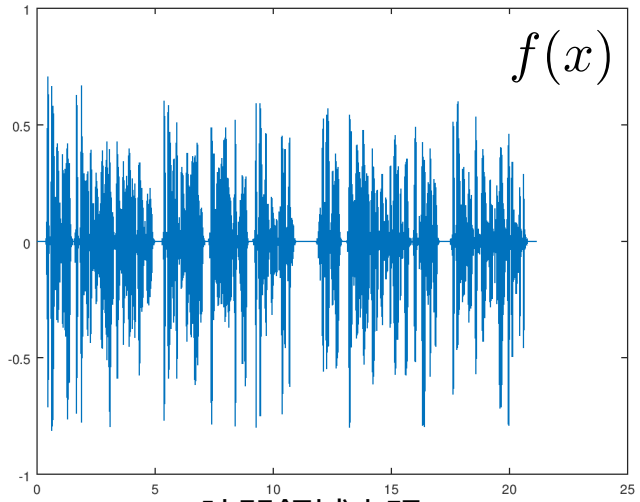
Fourier transform (2/2)　フーリエ変換　(2/2)

- （周期的）信号は周波数成分で表すことができる
- この考えを一般化すると，フーリエ変換は信号を周波数成分の分布として表現できる

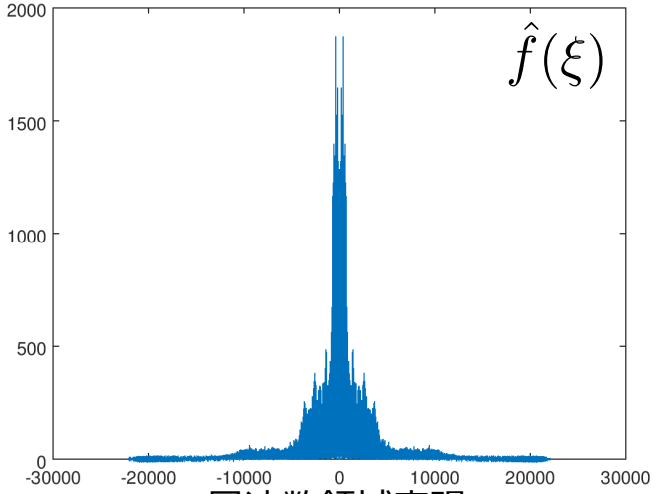
$$\hat{f}(\xi) := \int_{-\infty}^{\infty} f(x)\, e^{-2\pi i x \xi}\, dx$$

- The representation $f(\xi)$ in the frequency domain can be transformed back to that $f(x)$ in the temporal domain
- 周波数領域の表現$f(\xi)$は，時間領域の表現$f(x)$に変換して戻すことができる

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



時間領域表現



周波数領域表現

Computing frequency components
周波数成分の計算

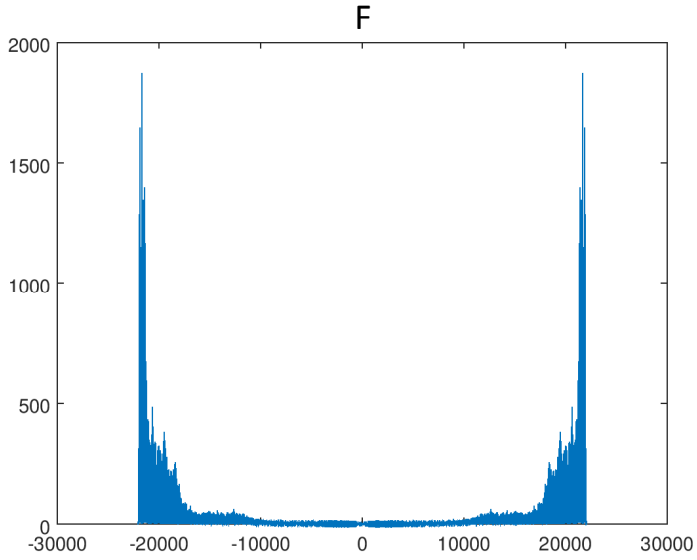
- `fft` performs Fourier transform (algorithm: *fast Fourier transform*) for an input signal

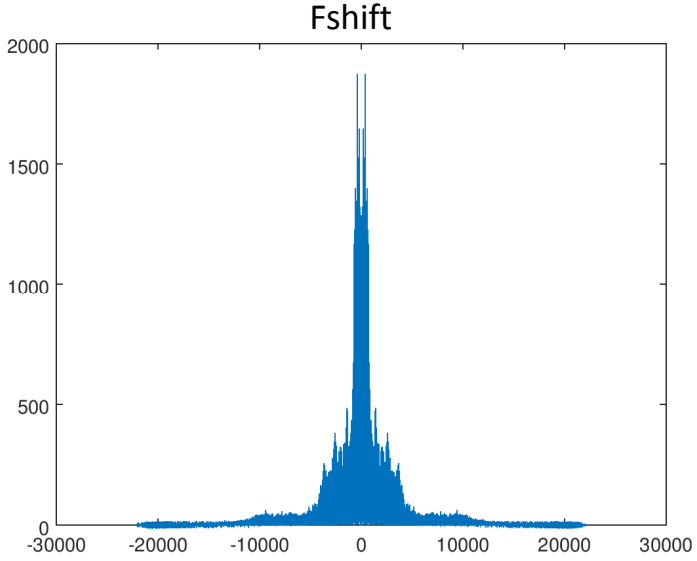
```
>> F=fft(f(:,1));
>> fs=size(F)
ans =
   932591
>> F(50000)
ans =  92.2299 +  1.2605i
>> df=freq/length(F);
>> xi=-freq/2:df:freq/2-df;
>> plot(xi,abs(F))
```

※`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





Computing frequency components
周波数成分の計算

- `fft`関数は，入力信号に対してフーリエ変換（高速フーリエ変換アルゴリズム）を行います。

```
>> F=fft(f(:,1));
>> fs=size(F)
ans =
   932591
>> F(50000)
ans =  92.2299 +  1.2605i
>> df=freq/length(F);
>> xi=-freq/2:df:freq/2-df;
>> plot(xi,abs(F))
```

F(5000)　Fの5000行目の値を表示

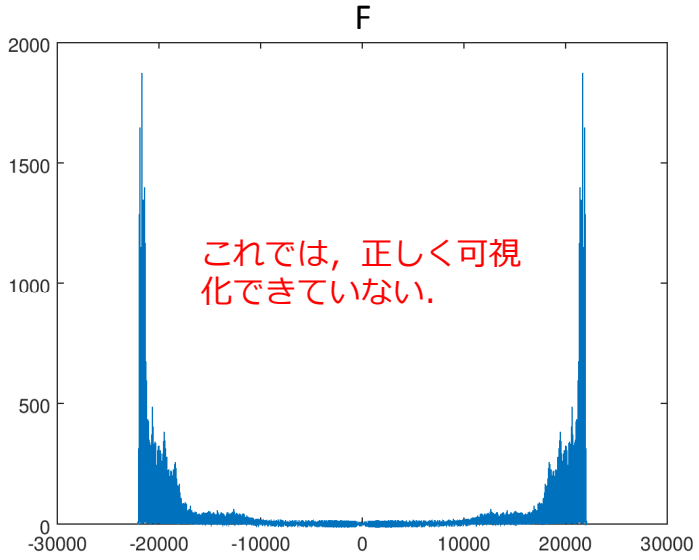
abs(x)：xの絶対値を出力する

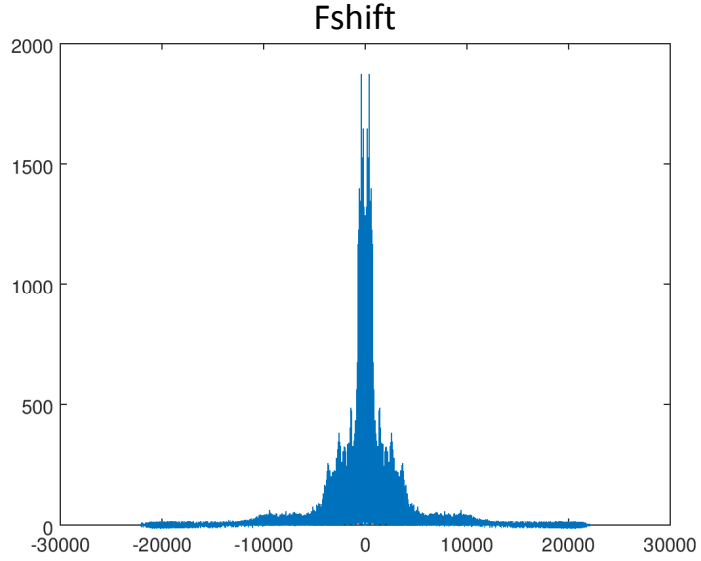
```
>> Fshift=fftshift(F);
>> plot(xi,abs(Fshift))
```

※`F`は，中心周波数ではなく両端にゼロ周波数成分があるベクトル．`fftshift`関数によって`F`をシフトして中心をゼロ周波数にします。 周波数成分は中央付近に集中していることが分かる．

※指定された範囲内の`freq/2`は、サンプリング周波数（ナイキスト周波数と呼ばれる）の1/2を超える周波数成分は正しくサンプリングできないという事実を反映しています。

fftshift(X)：ゼロ周波数成分を配列の中央に移動して，関数 fftの出力を再配置します．スペクトルの中心に，ゼロ周波数成分を配置することで，フーリエ変換の可視化が有効になります．

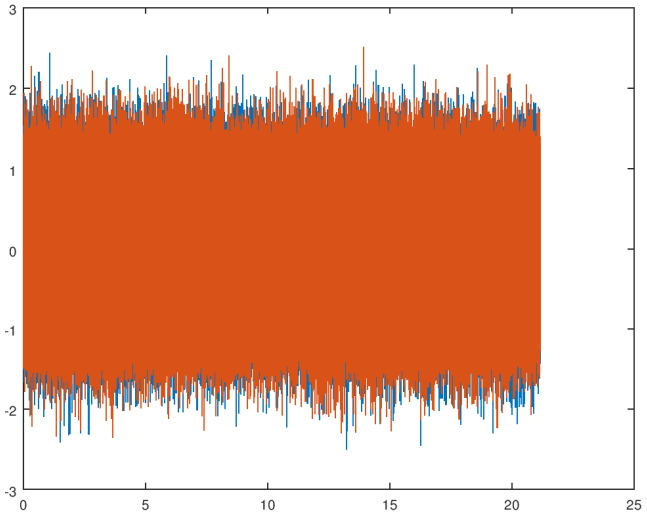




Noise reduction (1/2)　ノイズ低減処理(1/2)

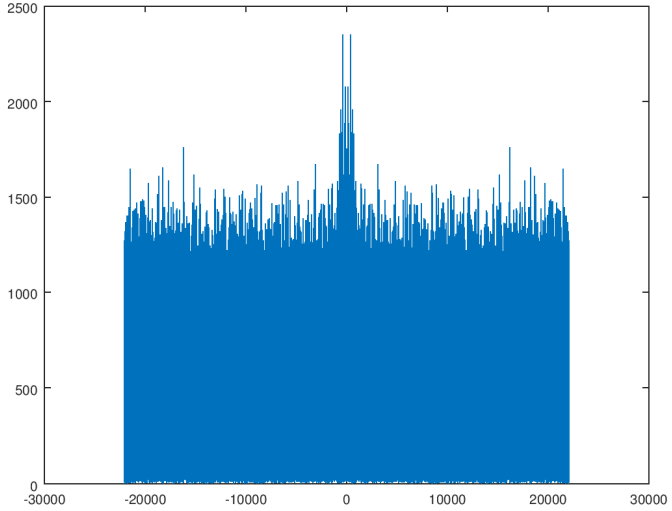
- Let’s adding noises to a signal intentionally
  信号にノイズを意図的に加えてみましょう！

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



- When seeing the signal in the frequency domain
  周波数領域で信号を見てみると・・・

```
>> F2shift=fftshift(fft(f2));
>> plot(xi,abs(F2shift(:,1)))
```



Noise reduction (2/2)　ノイズ低減処理(2/2)

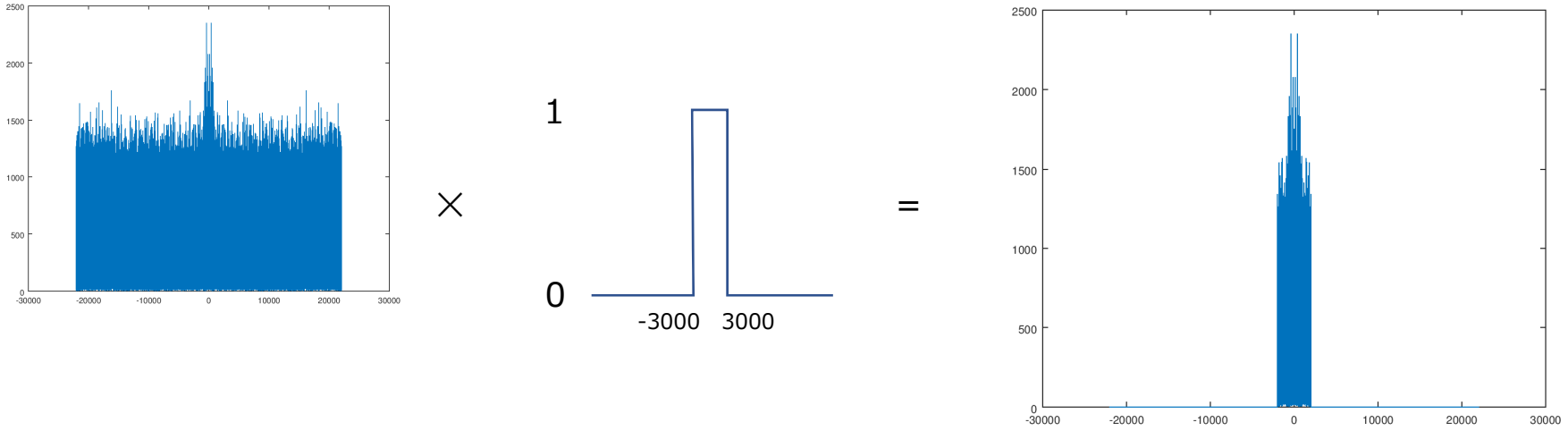
- Let’s remove high-frequency components and perform inverse Fourier transform
  - Creating a filter eliminating frequency components lower than 3kHz

```
>> 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)
```



Noise reduction (2/2)　ノイズ低減処理(2/2)

- 高周波成分を取り除いて逆フーリエ変換をしてみましょう！
  - 3kHz以下の周波数成分を除去するフィルタを作成する

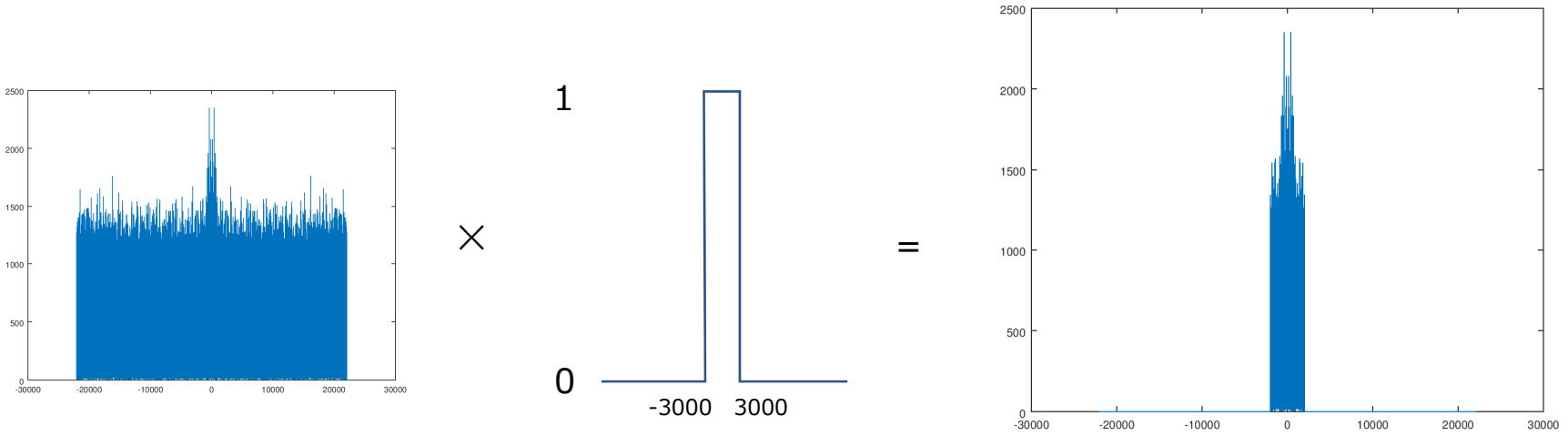
```
>> filter=abs(xi) < 3000;
```

abs(xi)<3000ならば１（真），そうでなければ０（偽）

※3kHzの値を変更した場合に結果がどのように変化するかを確認しよう！

  - 信号とフィルタの間の要素ごとの乗算を行い，次に，逆シフト（ifftshift）と逆フーリエ変換（ifft）を適用する.

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



Exercise 7.1　課題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
   さまざまな周波数の正弦波のデジタルデータを作成し、オーディオ信号として再生し、時間領域と周波数領域でプロットせよ
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
   作成した正弦波の1つを音声信号（test.wav）に加え，その結果得られる信号を時間領域と周波数領域にプロットせよ
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
   2. で得られた信号から加算された正弦波をできるだけ除去する方法を検討し，その結果を時間領域と周波数領域の信号のプロットで示せ