2018年4月9日更新

7. 信号処理

Signal processing

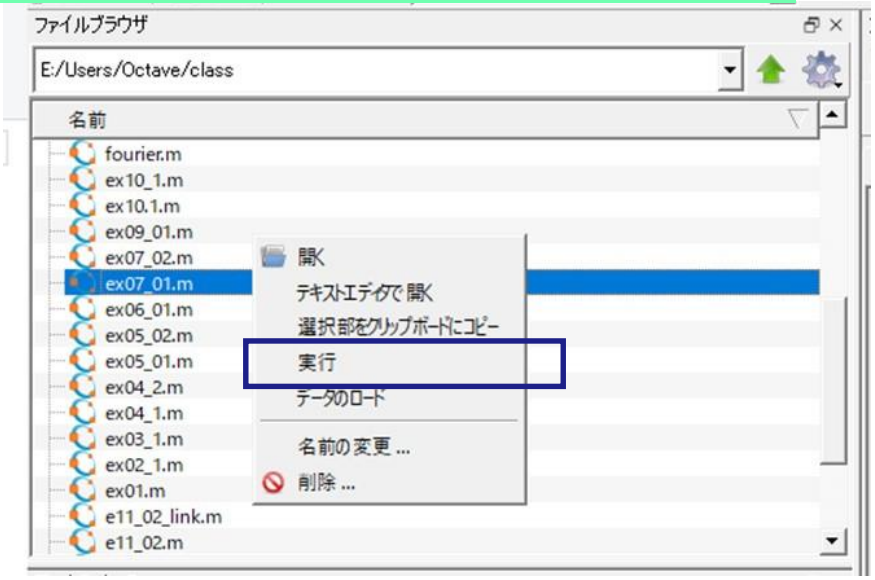
東北大学 大学院工学研究科
嶋田 慶太 shimada@m.tohoku.ac.jp

この副資料の場所:
http://www.pm.mech.tohoku.ac.jp/member/class

TOHOKU UNIVERSITY

知っていると便利な機能

マウス操作でファイルを実行



実行したいファイルの上で
「右クリック」▸「実行」

提出するファイルに対して
本当に動くか
確かめてください.

無限ループにはまってしまった (^^;

- ▸ コマンド上で，Ctrl + C で一時停止が可能

目次

- 信号の作成と基本操作
- 離散フーリエ変換
- 高速フーリエ変換

- オーディオデータの読み込み
- フーリエ変換
- ノイズ除去

(1) 純音の作成

横軸を時間，縦軸を信号強度として，
・時間区間 0 s から 1 s,
・サンプリング周波数10 kHz,
・信号の最大強度 10, 周波数10 Hz
の信号（sin）を作成しプロットせよ.

```
freq = 10000;
t = [0:1/freq:1]';
x = 10*sin(2*pi*10*t);
plot(t,x)
xlabel("Time [s]");
ylabel("Amplitude");
title("Sample 07 01")
```

◀ サンプリング周波数
◀ 横軸 t の設定
◀ 縦軸 の値の設定
◀ プロット
◀ X軸のタイトル
◀ Y軸のタイトル
◀ グラフのタイトル

連続関数も
適宜刻まなければ
コンピュータでは扱えない

補足

t: 列ベクトル 0 0.0001 0.0002 … 1 （第3回の内容; スペースの関係で行ベクトル）

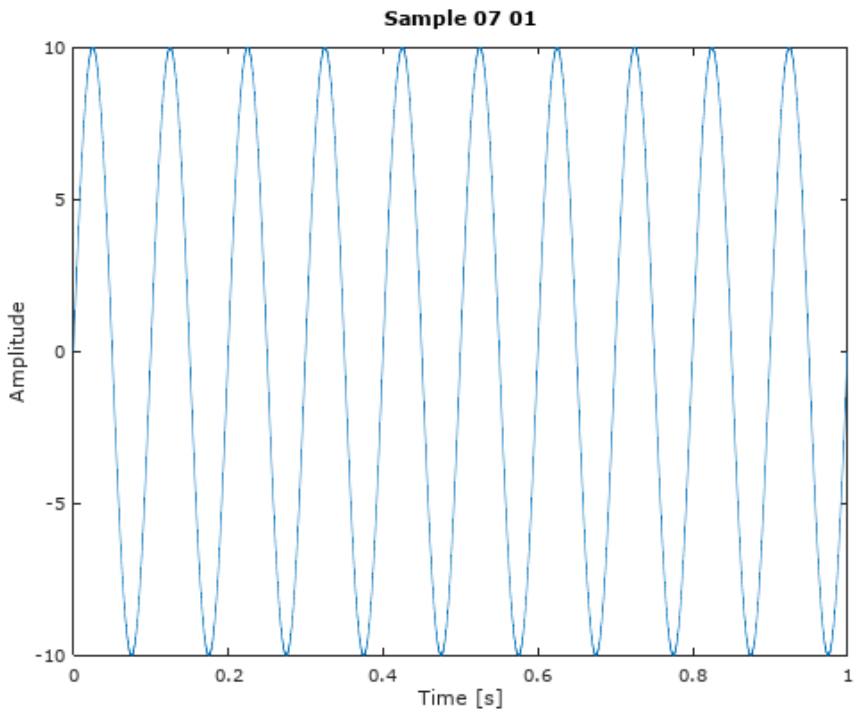
▼ 順次要素を`10*sin(2*pi*10*t)`に代入することに

x: 列ベクトル 0 0.0001 0.0002 … 1 （これも第3回の内容:Octave の要素ごと演算）

(1) 純音の作成

横軸を時間，縦軸を信号強度として，
・時間区間 0 s から 1 s,
・サンプリング周波数10 kHz,
・信号の最大強度 10, 周波数10 Hz
の信号（sin）を作成しプロットせよ.

```
freq = 10000;
t = [0:1/freq:1]';
x = 10*sin(2*pi*10*t);
plot(t,x)
xlabel("Time [s]");
ylabel("Amplitude");
title("Sample 07 01")
```



この時点での t と x は10001次元ベクトル

$$\{t_i\} = \begin{Bmatrix} 0 \\ 0.0001 \\ 0.0002 \\ \vdots \\ 1 \end{Bmatrix} \qquad \{x_i\} = \begin{Bmatrix} 0 \\ 0.062831 \\ 0.12566 \\ \vdots \\ 1 \end{Bmatrix}$$

例えば第2要素と第3要素は関連ありそうだが，ベクトルとしては独立した要素であるなぜなら，単位ベクトル $\mathbf{e}_i$ を i 番目の要素が 1 で他は 0 のベクトルとすると，

$$\{t_i\} = 0\cdot\mathbf{e}_1 + 0.0001\cdot\mathbf{e}_2 + 0.0002\cdot\mathbf{e}_3 + \ldots + 0\cdot\mathbf{e}_{10001}$$

$$\{x_i\} = 0\cdot\mathbf{e}_1 + 0.062831\cdot\mathbf{e}_2 + 0.12566\cdot\mathbf{e}_3 + \ldots + 1\cdot\mathbf{e}_{10001}$$

であり，当然ながら $\mathbf{e}_2$ を他の単位ベクトルの線形結合では表現できない．

(2) 複数信号の重ね合わせ

横軸を時間，縦軸を信号強度として，
・時間区間 0 s から 1 s,
・サンプリング周波数10 kHz,
・信号の最大強度 10, 15, 30, 周波数10, 25, 60 Hz
の重ね合わせた信号作成しプロットせよ.

```
freq = 10000;
t = [0:1/freq:1]';
x = sin(2*pi*t*[10,25,60])*[10;15;30];
plot(t,x)
xlabel("Time [s]");
ylabel("Amplitude");
title("Sample 07 02")
```

◀ サンプリング周波数
◀ 横軸 t の設定
◀ 縦軸 の値の設定
◀ プロット
◀ X軸，Y軸のタイトル
◀ グラフのタイトル

$$x = \sin(2\pi\{t_i\}(10,25,60))\begin{Bmatrix}10\\15\\30\end{Bmatrix}$$

となるので足し合わせになる

(2) 複数信号の重ね合わせ(別解)


```
freq = 10000;
t = [0:1/freq:1]';
x1 = 10*sin(2*pi*10*t);
x2 = 15*sin(2*pi*25*t);
x3 = 30*sin(2*pi*60*t);
x = x1 + x2 + x3;
plot(t,x)
xlabel("Time [s]");
ylabel("Amplitude");
title("Sample 07 02")
```

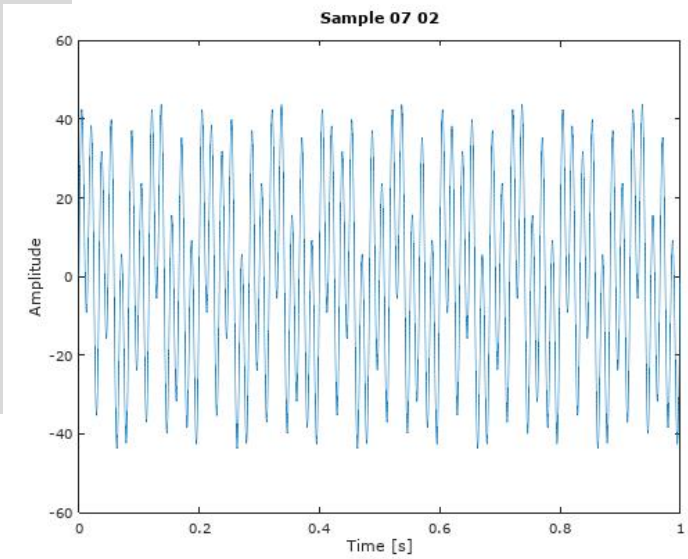
```
freq = 10000;
t = [0:1/freq:1]';
x = 10*sin(2*pi*10*t);
x += 15*sin(2*pi*25*t);
x += 30*sin(2*pi*60*t);
plot(t,x)
xlabel("Time [s]");
ylabel("Amplitude");
title("Sample 07 02")
```

これらでも構わないが，並列計算の関係上，行列を使ったほうが速いことが多い．
（これらだと大差がなかった）

(2) 複数信号の重ね合わせ

横軸を時間，縦軸を信号強度として，
・時間区間 0 s から 1 s,
・サンプリング周波数10 kHz,
・信号の最大強度 10, 15, 30, 周波数10, 25, 60 Hz
の重ね合わせた信号作成しプロットせよ.

```
freq = 10000;
t = [0:1/freq:1];
x = [10,15,30]*sin(2*pi*[10;25;60]*t);
plot(t,x)
xlabel("Time [s]");
ylabel("Amplitude");
title("Sample 07 02")
```

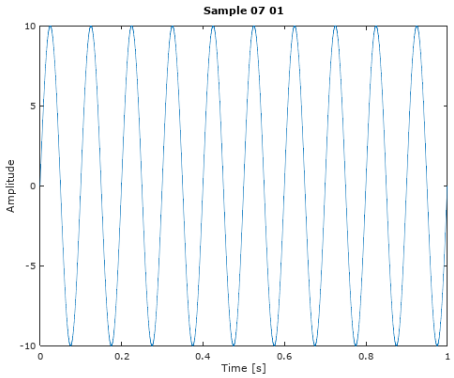


ここまでは適当な信号を作っただけ.

(3) 2つの純音信号の内積

横軸を時間，縦軸を信号強度として，
・時間区間 0 s から 1 s,
・サンプリング周波数10 kHz,
・信号の最大強度 10, 周波数10 Hz
の信号に対して，

(1) の信号



1, 2, 3, … 20 Hz，強度1の信号（sin, cos）を作成し内積を取れ.

```
freq = 10000;
t = [0:1/freq:1]';
x = 10 * sin(2*pi*10*t);
s = sin(2*pi*t*[1:20]);
c = cos(2*pi*t*[1:20]);
plot([1:20],s'*x,"o",[1:20],c'*x,"x")
xlabel("Frequency [Hz]");
ylabel("Intensity");
title("Sample 07 03")
```

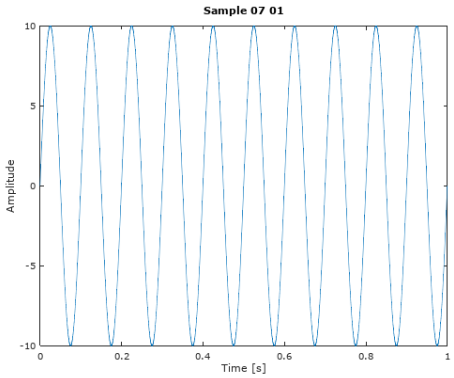
◀1～20 Hzの信号を作成

◀c, sが行ベクトル，x'が列ベクトルなので内積

2つの純音信号の内積

横軸を時間，縦軸を信号強度として，
・時間区間 0 s から 1 s,
・サンプリング周波数10 kHz,
・信号の最大強度 10, 周波数10 Hz
の信号に対して，

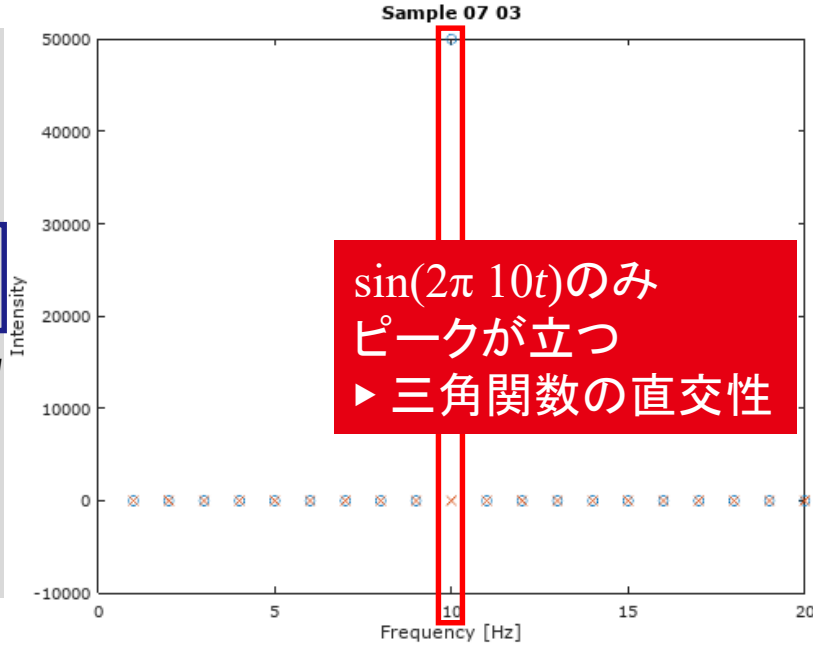
(1) の信号



1, 2, 3, … 20 Hz，強度1の信号（sin, cos）を作成し内積を取れ.

```
freq = 10000;
t = [0:1/freq:1]';
x = 10 * sin(2*pi*10*t);
s = sin(2*pi*t*[1:20]);
c = cos(2*pi*t*[1:20]);
plot([1:20],s'*x,"o",[1:20],c'*x,"x"
xlabel("Frequency [Hz]");
ylabel("Intensity");
title("Sample 07 03")
```

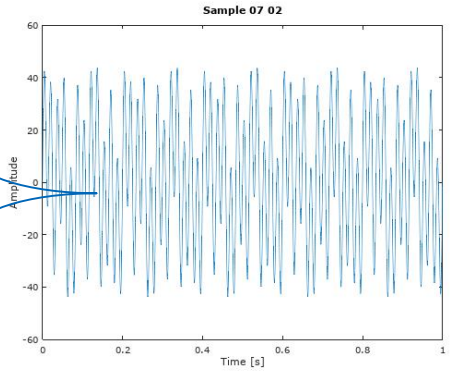
検査



(4) 重ね合わせ信号との内積

横軸を時間，縦軸を信号強度として，
・時間区間 0 s から 1 s,
・サンプリング周波数10 kHz,
・信号の最大強度 10, 15, 30, 周波数10, 25, 60 Hz
の重ね合わせた信号に対して，
1, 2, 3, … 80 Hz，強度1の信号（sin, cos）を作成し内積を取れ.

(2) の信号

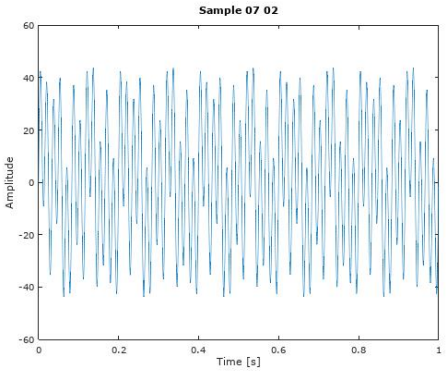



```
freq = 10000;
t = [0:1/freq:1]';
x = sin(2*pi*t*[10,25,60])*[10,15,30]';
s = sin(2*pi*t*[1:80]);
c = cos(2*pi*t*[1:80]);
plot([1:80],s'*x,"o",[1:80],c'*x,"x")
xlabel("Frequency [Hz]");
ylabel("Intensity");
title("Sample 07 04")
```

検査

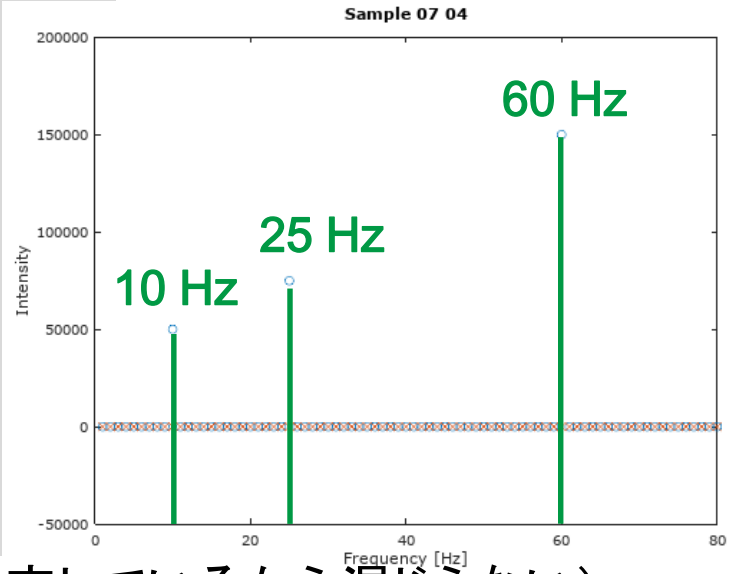
(4) 重ね合わせ信号との内積

横軸を時間，縦軸を信号強度として，
・時間区間 0 s から 1 s,
・サンプリング周波数10 kHz,
・信号の最大強度 10, 15, 30, 周波数10, 25, 60 Hz
の重ね合わせた信号に対して，
1, 2, 3, … 80 Hz，強度1の信号（sin, cos）を作成し内積を取れ.




```
freq = 10000;
t = [0:1/freq:1]';
x = sin(2*pi*t*[10,25,60])*[10,15,30]
s = sin(2*pi*t*[1:80]);
c = cos(2*pi*t*[1:80]);
plot([1:80],s'*x,"o",[1:80],c'*x,"x")
xlabel("Frequency [Hz]");
ylabel("Intensity");
title("Sample 07 04")
```

検査



▶ 内積を取る ▸ 成分を抜き出す（ほかの成分：直交しているから混じらない）

内積は成分を抜き出す操作

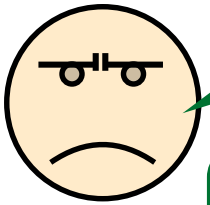
以上の性質は三角関数の直交性から説明される.

$$\mathbf{u}_f = \begin{Bmatrix} \exp\left(-\mathrm{i}\frac{2\pi}{N}f\times 0\right) \\ \exp\left(-\mathrm{i}\frac{2\pi}{N}f\times 1\right) \\ \vdots \\ \exp\left(-\mathrm{i}\frac{2\pi}{N}f(N-1)\right) \end{Bmatrix}$$

複素三角関数の点群

$$\mathbf{u}_f \cdot \mathbf{u}_\varphi = \sum_{t=0}^{N-1} \exp\left(-\mathrm{i}\frac{2\pi}{N}(f-\varphi)t\right) = N\delta_{f\varphi}$$

内積 ▸ クロネッカーのデルタ



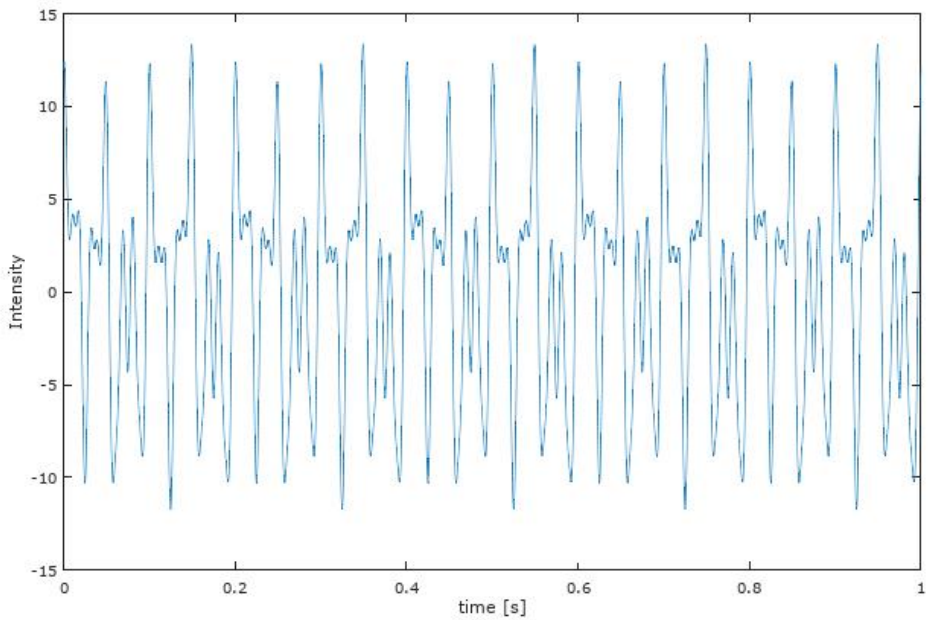
なんで急に複素三角関数？

sin と cos の両方の情報があってわかりやすいから…わかりにくければ分解してください

内積を取る ▸ 周波数成分を抜き出す（他の成分は 0 になる）

いかなる信号に対しても有効 ▸

信号の周波数解析（離散フーリエ変換:DFT）

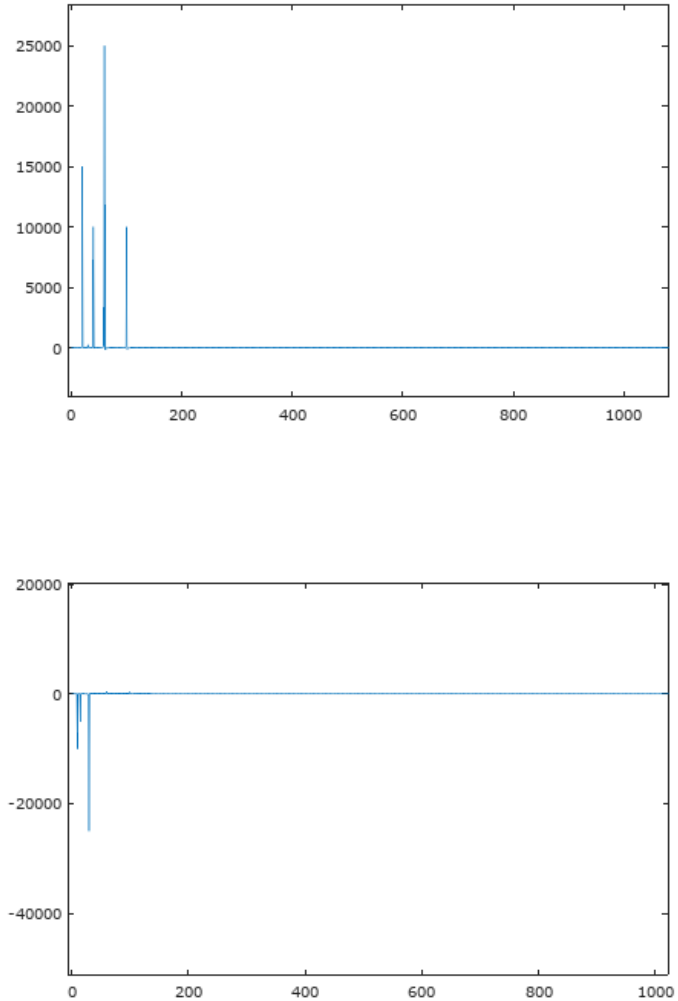


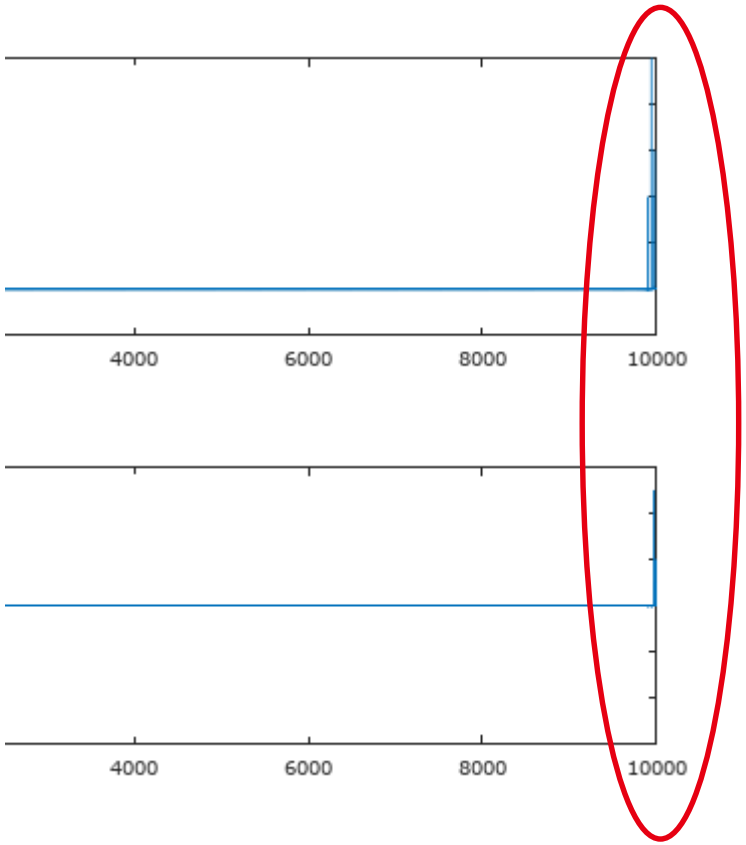
・時間区間 0 s から 1 s,
・サンプリング周波数10 kHz
で得られた信号である.
この信号について周波数解析せよ.

```
load sample.txt
N = length(x);
ttl_time = (N-1)/freq;
t = linspace(0,ttl_time,N);
f = linspace(0,freq-freq/N,N);
X = x * exp(-i*2*pi*f'*t);
subplot(2,1,1); plot(f, real(X));
subplot(2,1,2); plot(f, imag(X));
```

◀上記の信号の強度xと周波数freqを格納したファイルを読み込む
◀length: 長さ(要素数)を数える関数(行・列の方で長い方を出す)
◀要素数とサンプリング周波数から総時間を計算(今回は1 s)
◀linspace(a,b,n): aからbまで等間隔にn個に区切った行ベクトルを生成する.
◀xと周波数信号の内積をXに格納
◀cos成分の表示
◀sin成分の表示

解析完了・・・ただし・・・





高周波域にも
信号が…？

これには数学的な理由が

$N = 8$ の場合

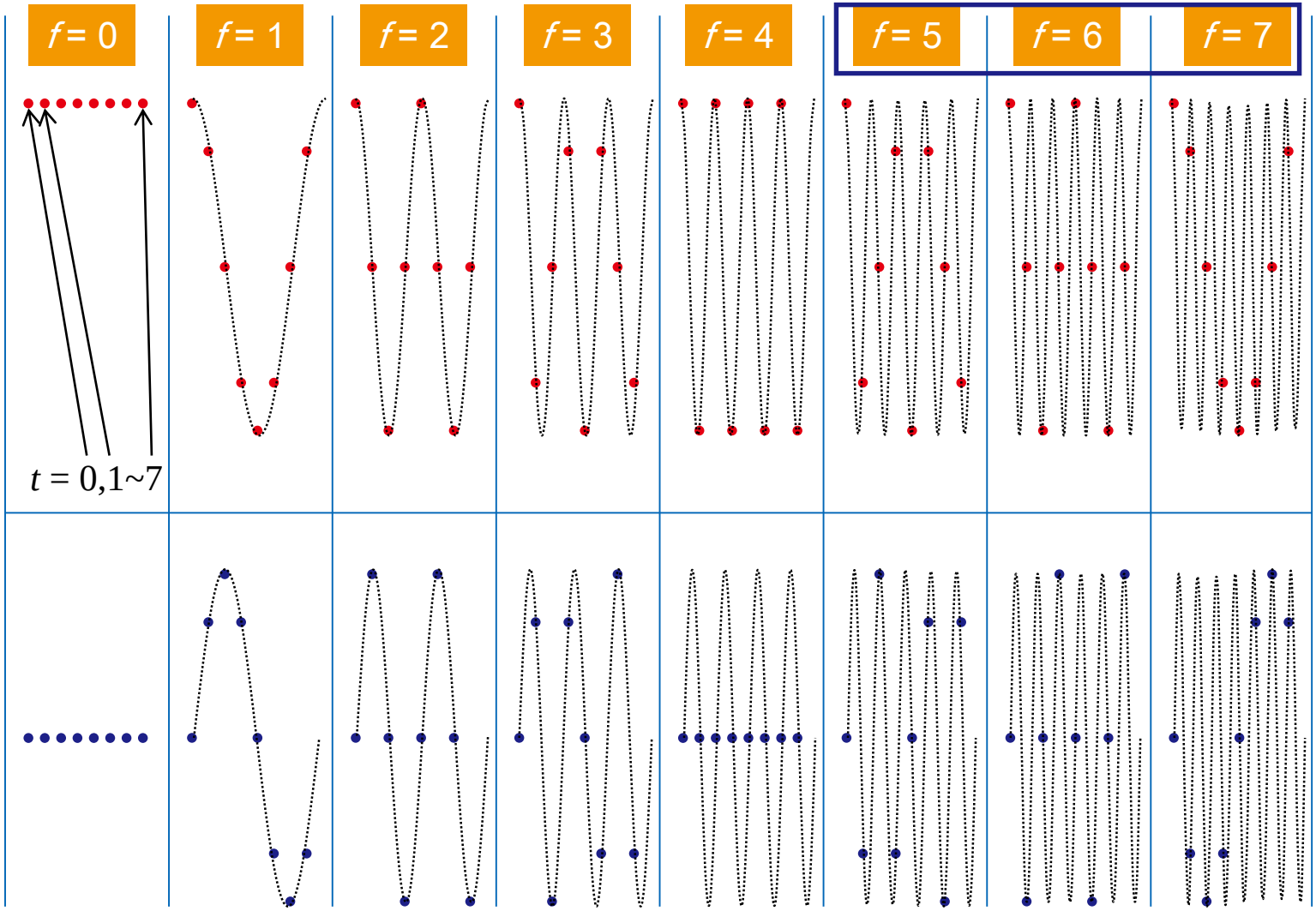
再現できない ▸ シャノンのサンプリング定理

$$\exp\left(-\mathrm{i}\,\frac{2\pi f t}{N}\right)$$

に関して

cos

sin

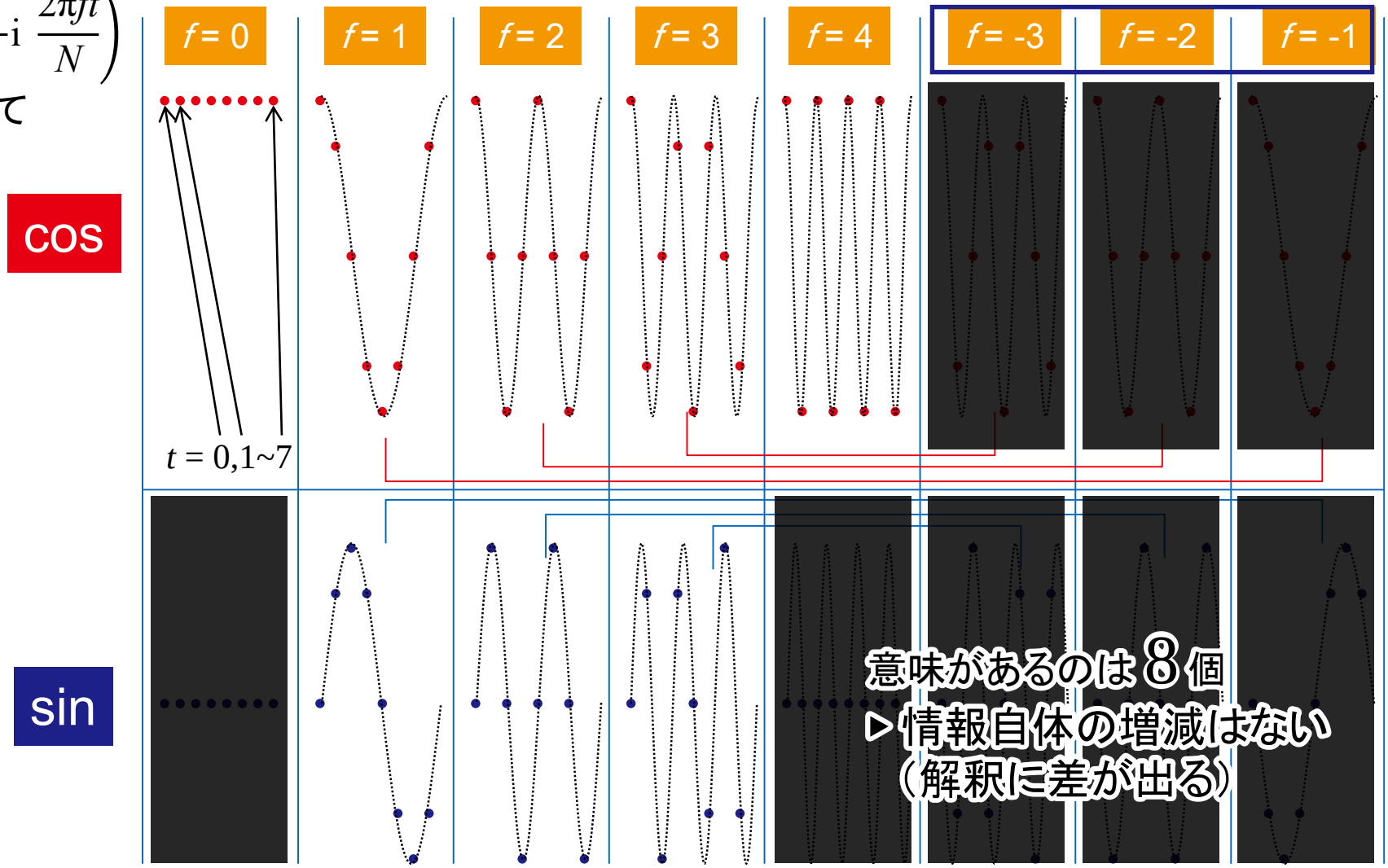


$N = 8$ の場合

結果的に符号反転の状態となる.

$$\exp\left(-\mathrm{i}\,\frac{2\pi f t}{N}\right)$$

に関して



高速フーリエ変換（FFT）

FFT: 離散フーリエ変換を高速化したアルゴリズム

```
load sample.txt
N = length(x);
ttl_time = (N-1)/freq;
t = linspace(0,ttl_time,N);
f = linspace(0,freq-freq/N,N);
tic;
X_dft = x * exp(-i*2*pi*f'*t);
printf("DFT took %f seconds¥n",toc)
tic;
X_fft = fft(x);
printf("FFT took %f seconds¥n",toc)
subplot(2,1,1); plot(f, real(X_dft), f,
real(X_fft),"o");
subplot(2,1,2); plot(f, imag(X_dft), f,
imag(X_fft),"o");
```

tic: タイマーを仕掛け，
toc: 掛かった時間を表示

tic: タイマーを仕掛け，
toc: 掛かった時間を表示

fft: 高速フーリエ変換

DFT took 10.468022 seconds
FFT took 0.000499 seconds

▶ FFTは本当に速い

数学的な説明について

もう少し数学的な説明がほしい場合は
別添資料を読んでください.

実践編

オーディオデータの読み込み

▶ 音声信号の読み取り: audioread

```
>> [x,freq]=audioread('test.wav');
```

x: 音声の強度信号
　列ベクトルに各チャンネルの信号を格納
　（1ch. ▸モノラル・2ch. ▸ステレオ）

freq: サンプリング周波数 ▸ 逆数: サンプリング間隔

▶ 音声信号の再生: sound

```
>> sound(x,freq);
```

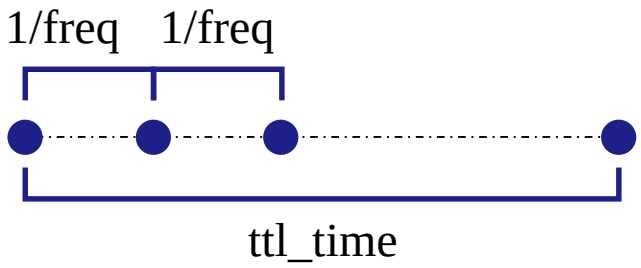
そのまま再生

```
>> sound(x,0.7*freq);
```

周波数を落として再生

横軸(時間軸)の設定


```
>> N = rows(x);
>> ttl_time = (N-1)/freq;
```



植木算を考え，データ点数から1引く
（間隔は(n - 1)個）

N: データ点数　行数を数える関数: `rows`
（列数を数える関数は `columns`）

ttl_time: サンプリングしたトータルの時間

```
>> t=linspace(0,ttl_time,N)';
```

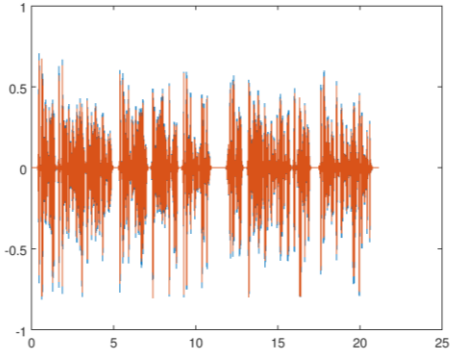
xと合わせるために転置
（しなくてもグラフは描ける）

linspace(start, goal, n) : start から goal までを等間隔に区切った n 列の行ベクトル

```
>> plot(t,x)
```

横軸: 時間

縦軸: 強度



音声信号の例
（実際のtest.wavは全く違う）

高速フーリエ変換

高速フーリエ変換: fft

```
>> X = fft(x);
```
◀今回はFFT後の値を大文字変数に入れる

FFT後の各要素

```
>> size(X)
ans =
   211289 1
>> X(50000)
ans = -5.5782 + 3.2176i
```
◀変換前と同じ要素数

◀例えば50000番のデータは

◀cos 成分と sin 成分（強度と位相に関係）

DFTとの比較

```
>> exp(-i*2*pi*(50000-1)/N*[0:211288])*x
ans = -5.5782 + 3.2176i
```

対称位置のデータ

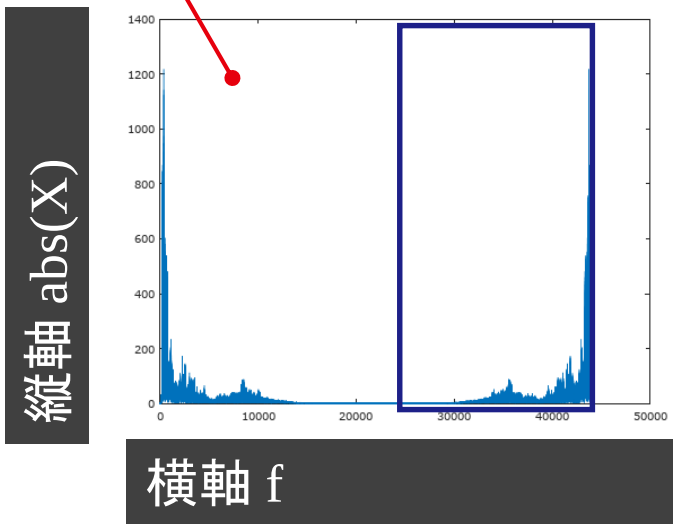
```
>> X(N-(50000-1)+1)
ans = -5.5782 - 3.2176i
```
◀虚部（sin 成分）のみ反転

スライドNo.16-17の説明

FFT後の横軸(周波数軸)とFFTシフト

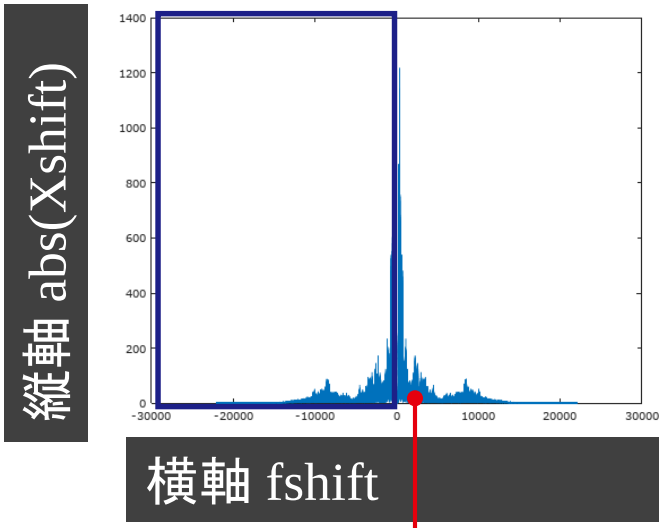

```
>> f=linspace(0,freq-freq/N,N);
>> plot(f,abs(X))
```

複素数の絶対値(**Abs**olute value)
ちなみに偏角は arg



fftshift

中心で切り取って
平行移動



データを軸対称にする: fftshift

```
>> Xshift=fftshift(X);
>> fshift=f-(floor(N/2))*freq/N;
>> plot(fshift,abs(Xshift))
```

フーリエ変換のまとめ（plotやsoundを省略すると…）


```
[x,freq]=audioread('test.wav');
N=rows(x);
ttl_time=(N-1)/freq;
t=linspace(0,ttl_time,N)';
X = fft(x(:,1));
f=linspace(0,freq-freq/N,N)';
Xshift=fftshift(X);
fshift=f-(floor(N/2))*freq/N;
```

- データ読み込み
- データ点数
- 総時間の計算
- 時間領域での横軸生成
- FFT解析し，Xに代入
- 周波数領域での横軸
- f = 0 を中心にシフト（縦軸）
- f = 0 を中心にシフト（横軸）

	時間領域	周波数領域（f = 0から始まる）	f = 0を中心に移動
縦軸 横軸	x: 時間領域での信号強度 t: 時間軸	X: 周波数領域での信号強度 f: 周波数軸	Xshift fshift

共通情報	N: サンプル数 freq: サンプリグ周波数 ttl_time: 総時間

横軸の終端に関連

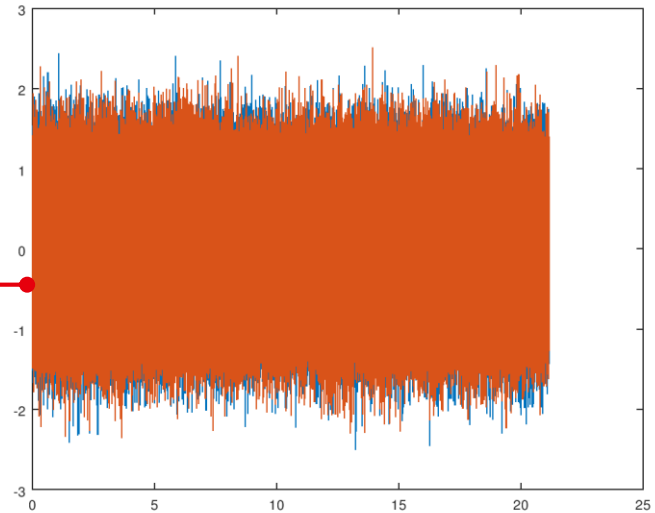
ノイズ除去 (1/3)

乱数列を用いてあえてノイズを入れる

```
>> x_n=x+0.5*randn(size(x));
```

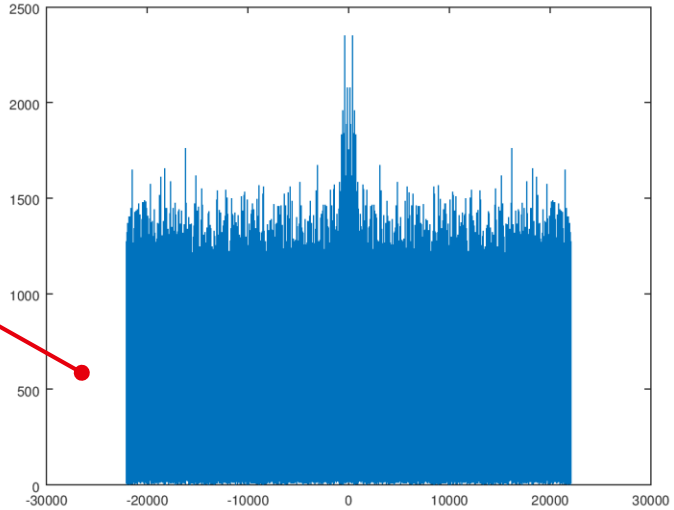
↳ ノイズ入り信号x_n: xと同サイズの乱数を作り重畳

```
>> sound(x_n,freq);

>> plot(t, x_n);
```



周波数領域で見ると,

```
>> X_n_shift=fftshift(fft(x_n));

>> lot(fshift,abs(X_n_shift(:,1)))
```



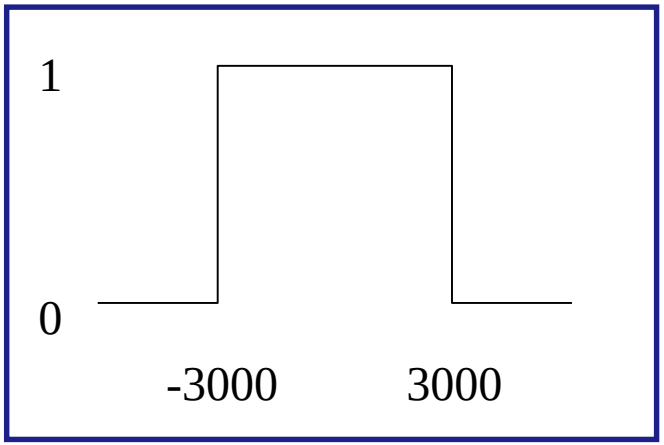
ノイズ除去 (2/3)

高周波成分を取り除き，逆フーリエ変換 (`ifft`) する

▶ フィルター変数を作る

```
>> filt = abs(fshift) < 3000;
```

不等式も「要素ごと計算」をする．
条件が真なら1，偽なら0を返す．



fshift	fshift<3000
2.9991e+003	1
2.9993e+003	1
2.9995e+003	1
2.9997e+003	1
2.9999e+003	1
3.0001e+003	0
3.0003e+003	0

※教科書のfilter も変数として使っているが，
同名の関数があるので，混同を避けるため名前を変えている．
Octaveでは予約語にも代入「できてしまう」ので，予期せぬトラブルを生じることがある．

ノイズ除去 (3/3)

逆FFT: `ifft`

```
>> x_n_filt = ifft(ifftshift(X_n_shift.*filt));
```

「要素ごと積」により，3000 Hz外の値は0に

- `fftshift`のシフトを元に戻す．

```
>> sound(x_n_filt,freq)
```

フィルター後の音声を聞いてみよう

ノイズ除去のまとめ


```
x_n=x+0.5*randn(size(x));
X_n_shift=fftshift(fft(x_n));
filt = abs(fshift) < 3000;
x_n_filt = ifft(ifftshift(X_n_shift.*filt'));
```

時間領域

縦軸
横軸

x: 時間領域での信号強度
t: 時間軸

x_n: xにノイズを付加

x_n_filt: filter後の信号

FFT

IFFT

周波数領域

X_n_shift: x_nのFFT（さらにshift）

filter:
3000 Hz 以内は1,
それ以外は0
という変数

課題 7.1

1. 自分の学籍番号の4桁の数字部分をXとし，X Hzの信号Qを test.wav に重畳した信号をQ2とするときの，時間領域，周波数領域での信号Q2のグラフを描け.
2. Q2からノイズをできるだけ取り除くことのできるプログラムを考え，ノイズ除去後の信号Q3の時間領域，周波数領域での信号をグラフに描け.

! 教科書と少し変えています

※ノイズ除去はQ2しか見えない状態である，と仮定して除去するプログラムを考えること．すなわち「Q2-Q」ではダメ.
（Q2だけを見てQを抽出できるプログラムがあれば別だが．）

おまけ 最小二乗法との関係

先週は一次関数で近似するため，モデル関数を $ax+b$ の形式とした．

$$\{\varepsilon_i\} = \{y_i\} - [\{x_i\}\{1\}]\begin{Bmatrix} a \\ b \end{Bmatrix} \qquad \boxed{\begin{Bmatrix} a \\ b \end{Bmatrix} = (X^T X)^{-1} X^T \{y_i\}}$$

モデル関数を三角関数の級数として表現し，

$$X = \left[\left\{\cos\frac{2\pi}{N}\cdot 0\cdot x\right\}\cdots\left\{\cos\frac{2\pi}{N}\cdot\left\lceil\frac{N-1}{2}\right\rceil\cdot x\right\}\left\{\sin\frac{2\pi}{N}\cdot 1\cdot x\right\}\cdots\left\{\sin\frac{2\pi}{N}\cdot\left\lfloor\frac{N-1}{2}\right\rfloor\cdot x\right\}\right] \qquad x = \begin{Bmatrix} 0 \\ 1 \\ \vdots \\ N \end{Bmatrix}$$

それぞれのベクトルが互いに直交するので $X^T X$ が対角化

- 正規化すれば X^T を $\{y_i\}$ に掛けるだけ．
- それがまさにフーリエ変換

ただし，データが等間隔に並んでいることが前提．

ヒントと提出について

グラフを二つ以上書く場合には

```
subplot(i,j,k)
```

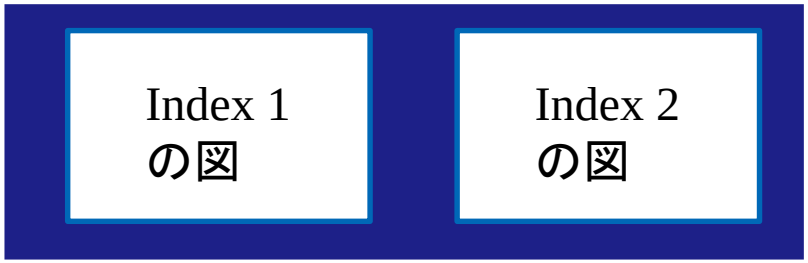
というコマンドが使える.

▲ 行　▲ 列　▲ インデックス

例

```
subplot(1,2,1) plot(~~~
subplot(1,2,2) plot(~~~
```

1行2列で図を貼る.



- 提出するファイルはtest.wavというファイルが作業ディレクトリに存在すると仮定して作成してよい.

- test.wavを添付しないでよい.（というか，添付してはいけない.）