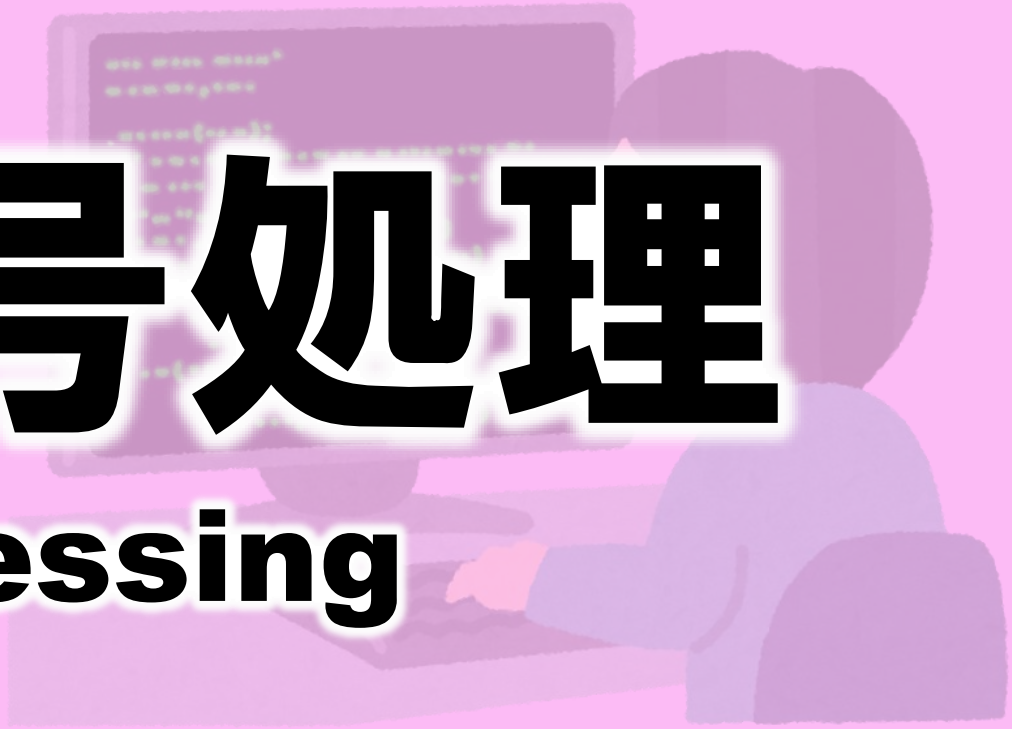
2019年7月1日更新



7. 信号処理

Signal processing

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

TOHOKU UNIVERSITY

TOHOKU
UNIVERSITY

オーディオデータの読み込み

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);
```

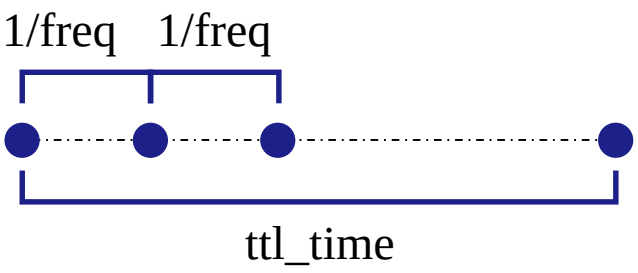
そのまま再生

```
>> 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

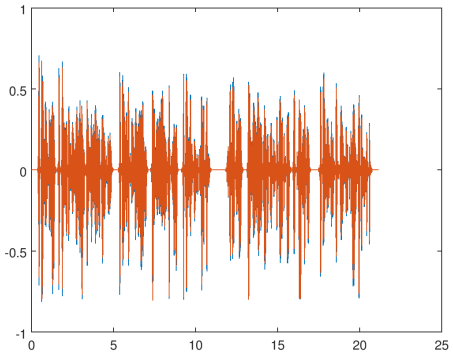
```
>> t=linspace(0,ttl_time,N)';
```

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）

この授業での列ベクトルと行列の表現(2)

この授業での行列・列ベクトルの表現

列 ベクトル

▶ 波括弧{}で表現

*世界的にこれがスタンダードというわけではないので
もし別の機会で使おうと思ったら，ルールを書いてください．

誤解がない場合は

$\begin{Bmatrix} x_1 \\ \vdots \\ x_n \end{Bmatrix}$ → 縮記 → $\{x_i\}_{i=[1,\,n]}$ → 縮記 → $\{x_i\}$

例えば

$\{1\}$ $\{0\}$

は全要素が
1，0の列ベクトル

* $\boldsymbol{X}$のように太字もよく使う．
* 行ベクトルは$\{x_i\}^T$のように列ベクトルの転置で表す

行列

▶ 大文字もしくは角括弧[]で表現

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$$

例えば

$[\{x_i\}\{y_i\}]$

は i 行の成分がx_i, y_iの $n \times 2$ 行列

関数と関連した列ベクトルの表現

0からはじまり，
1ずつ増える
N次元列ベクトル

$$\begin{Bmatrix} 0 \\ 1 \\ \vdots \\ N-1 \end{Bmatrix} = \{i\}$$

今回は0から
であることに注意

◂を用いた表現

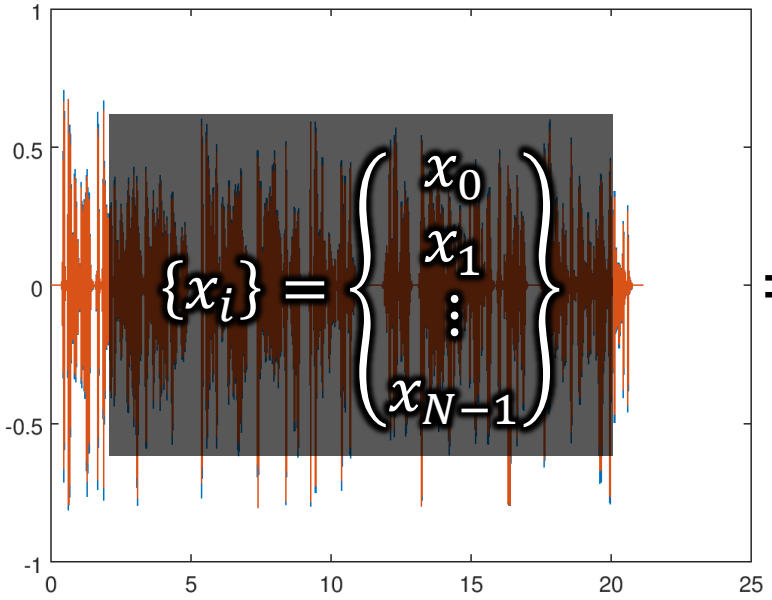
$$\begin{Bmatrix} \cos\left(\frac{2\pi}{N}\cdot 0\right) \\ \cos\left(\frac{2\pi}{N}\cdot 1\right) \\ \vdots \\ \cos\left(\frac{2\pi}{N}\cdot (N-1)\right) \end{Bmatrix} = \cos\left(\frac{2\pi}{N}\{i\}\right)$$

Octaveの引数と同様に，引数部分をベクトルで表し，
全体もベクトルで表されることとする．

TOHOKU UNIVERSITY

離散フーリエ変換（DFT）

信号は波の重ね合わせ（証明はしないけど）



N次元ベクトルとして
格納されているデータ

$=$

$$\frac{a_0}{2}\cos\left(\frac{2\pi}{T}\{t_i\}\cdot 0\right)$$
$$+a_1\cos\left(\frac{2\pi}{T}\{t_i\}\cdot 1\right)$$
$$+a_2\cos\left(\frac{2\pi}{T}\{t_i\}\cdot 2\right)$$
$$\vdots$$

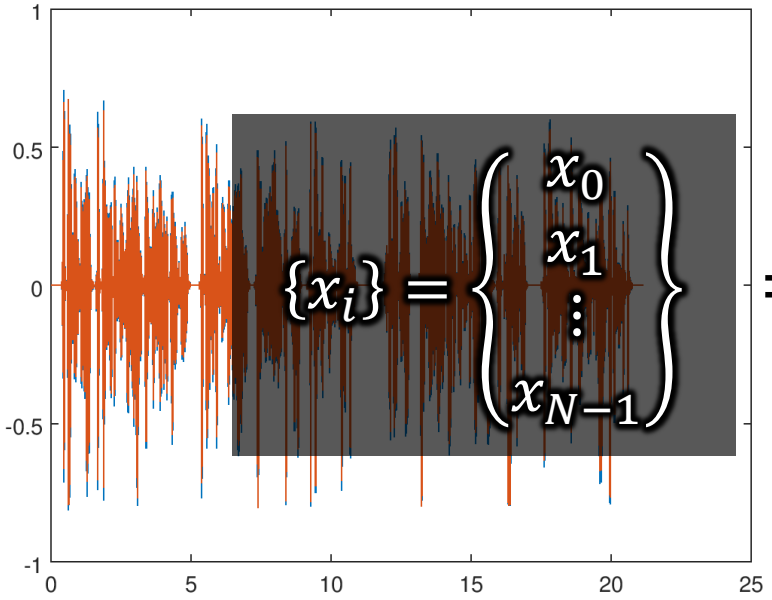
$$+b_1\sin\left(\frac{2\pi}{T}\{t_i\}\cdot 1\right)$$
$$+b_2\sin\left(\frac{2\pi}{T}\{t_i\}\cdot 2\right)$$
$$\vdots$$

N: データ点数
f: サンプリング周波数
T: 総時間 $(= (N/f))$

$$\{t_i\} = \begin{Bmatrix} 0 \\ 1/f \\ 2/f \\ \vdots \\ (N-1)/f \end{Bmatrix}$$

という形式で表現できる.（分解できる）
また式をよくみると時間の要素はキャンセルでき,

信号は波の重ね合わせ（証明はしないけど）



$$= \frac{a_0}{2}\cos\left(\frac{2\pi}{N}\{i\}\cdot 0\right) + a_1\cos\left(\frac{2\pi}{N}\{i\}\cdot 1\right) + a_2\cos\left(\frac{2\pi}{N}\{i\}\cdot 2\right) + \cdots + b_1\sin\left(\frac{2\pi}{N}\{i\}\cdot 1\right) + b_2\sin\left(\frac{2\pi}{N}\{i\}\cdot 2\right) + \cdots$$

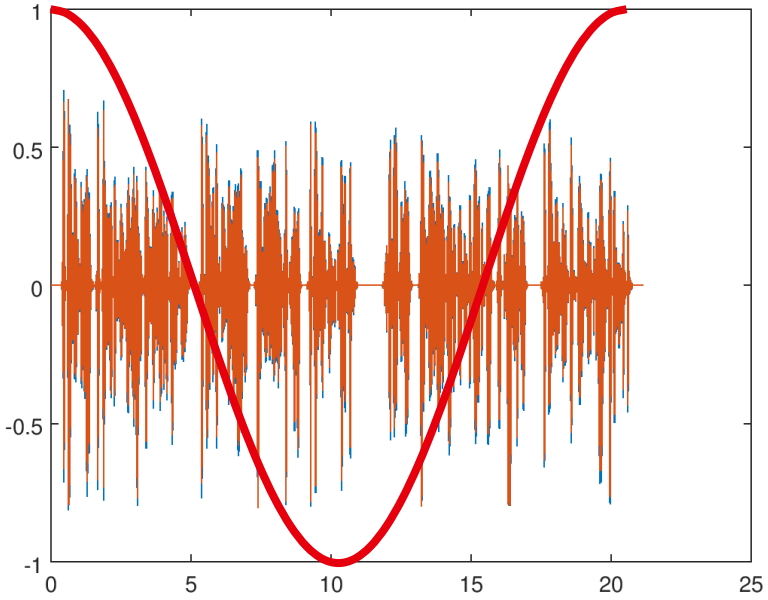
N: データ点数

$$\{i\} = \begin{Bmatrix} 0 \\ 1 \\ \vdots \\ (N-1) \end{Bmatrix}$$

という表現できる.

あとは係数 a_i b_i が分かれば分解完了

内積を取る ▶ 成分の抜出し



$$\frac{a_0}{2}\cos\left(\frac{2\pi}{N}\{i\}\cdot 0\right)$$
$$+a_1\cos\left(\frac{2\pi}{N}\{i\}\cdot 1\right)$$
$$+a_2\cos\left(\frac{2\pi}{N}\{i\}\cdot 2\right)$$
$$\vdots$$

$$+b_1\sin\left(\frac{2\pi}{N}\{i\}\cdot 1\right)$$
$$+b_2\sin\left(\frac{2\pi}{N}\{i\}\cdot 2\right)$$
$$\vdots$$

たとえば

$$\cos\left(\frac{2\pi}{N}\{i\}\cdot 1\right)$$

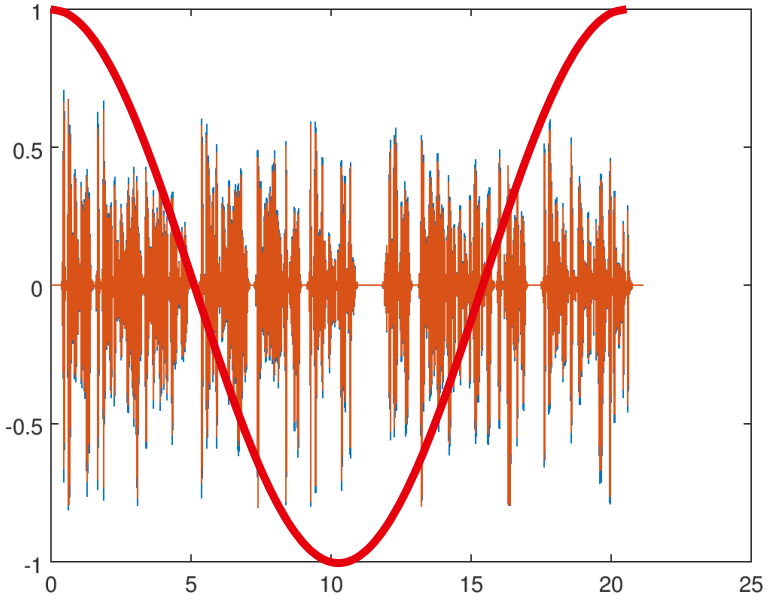
がどのくらい入っているかを調べたい

▶ 内積を計算

$$\sin\alpha\cos\beta=\frac{1}{2}[\sin(\alpha+\beta)+\sin(\alpha-\beta)]$$
$$\sin\alpha\sin\beta=\frac{1}{2}[\cos(\alpha+\beta)-\cos(\alpha-\beta)]$$
$$\cos\alpha\cos\beta=\frac{1}{2}[\cos(\alpha+\beta)+\cos(\alpha-\beta)]$$

中身が一致しない場合は波のまま▸総和を取れば0

内積を取る ▶ 成分の抜出し



$$\frac{a_0}{2}\cos\left(\frac{2\pi}{N}\{i\}\cdot 0\right)$$

$$+a_1\cos\left(\frac{2\pi}{N}\{i\}\cdot 1\right)$$

$$+a_2\cos\left(\frac{2\pi}{N}\{i\}\cdot 2\right)$$

$$\vdots$$

$$+b_1\sin\left(\frac{2\pi}{N}\{i\}\cdot 1\right)$$

$$+b_2\sin\left(\frac{2\pi}{N}\{i\}\cdot 2\right)$$

$$\vdots$$

▶ 内積を計算

$$\frac{N}{2}a_1$$

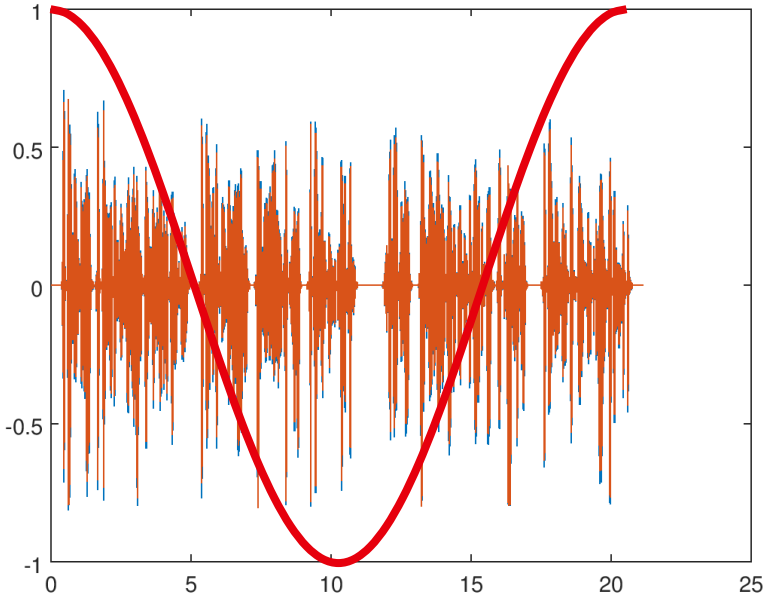
が残る.

たとえば

$$\cos\left(\frac{2\pi}{N}\{i\}\cdot 1\right)$$

がどのくらい入っているかを調べたい

内積を取る ▶ 成分の抜出し



もともと，この総和は

$$\{x_i\} = \begin{Bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{Bmatrix}$$

なので，これとの内積でも

▶ 内積を計算

たとえば

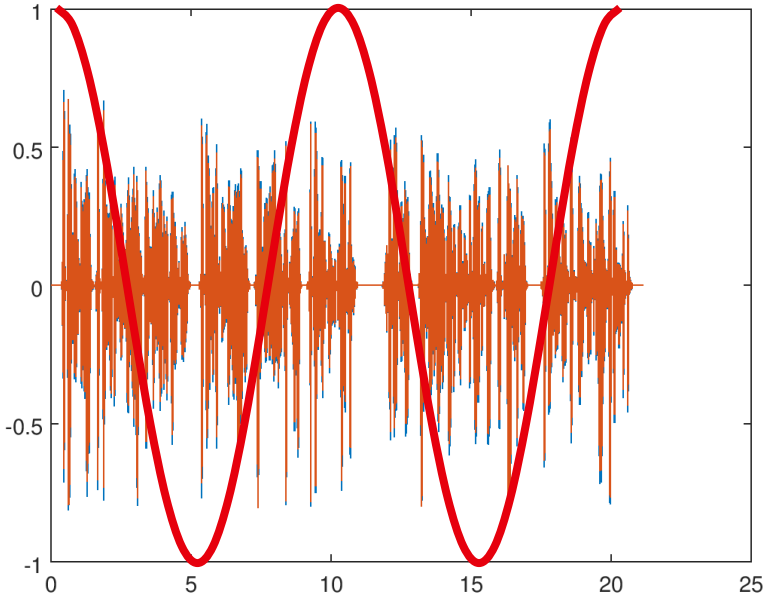
$$\cos\left(\frac{2\pi}{N}\{i\}\cdot 1\right)$$

がどのくらい入っているかを調べたい

$$\frac{N}{2}a_1$$

が残る．

内積を取る ▶ 成分の抜出し



$$\frac{a_0}{2}\cos\left(\frac{2\pi}{N}\{i\}\cdot 0\right)$$
$$+a_1\cos\left(\frac{2\pi}{N}\{i\}\cdot 1\right)$$
$$+a_2\cos\left(\frac{2\pi}{N}\{i\}\cdot 2\right)$$
$$\vdots$$

$$+b_1\sin\left(\frac{2\pi}{N}\{i\}\cdot 1\right)$$
$$+b_2\sin\left(\frac{2\pi}{N}\{i\}\cdot 2\right)$$
$$\vdots$$

たとえば

$$\cos\left(\frac{2\pi}{N}\{i\}\cdot 2\right)$$

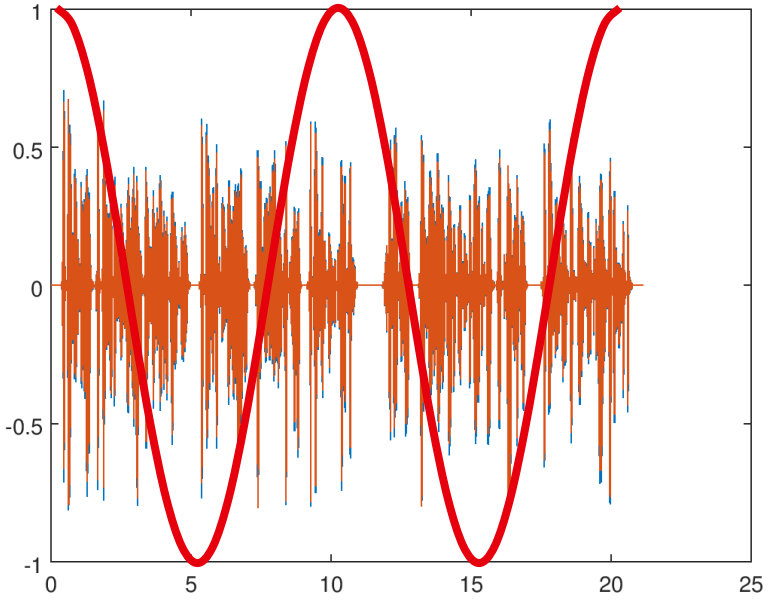
がどのくらい入っているかを調べたい

▶ 内積を計算

$$\sin\alpha\cos\beta=\frac{1}{2}[\sin(\alpha+\beta)+\sin(\alpha-\beta)]$$
$$\sin\alpha\sin\beta=\frac{1}{2}[\cos(\alpha+\beta)-\cos(\alpha-\beta)]$$
$$\cos\alpha\cos\beta=\frac{1}{2}[\cos(\alpha+\beta)+\cos(\alpha-\beta)]$$

中身が一致しない場合は波のまま▸総和を取れば0

内積を取る ▶ 成分の抜出し



$$\frac{a_0}{2}\cos\left(\frac{2\pi}{N}\{i\}\cdot 0\right)$$

$$+a_1\cos\left(\frac{2\pi}{N}\{i\}\cdot 1\right)$$

$$+a_2\cos\left(\frac{2\pi}{N}\{i\}\cdot 2\right)$$

$$\vdots$$

$$+b_1\sin\left(\frac{2\pi}{N}\{i\}\cdot 1\right)$$

$$+b_2\sin\left(\frac{2\pi}{N}\{i\}\cdot 2\right)$$

$$\vdots$$

▶ 内積を計算

$$\frac{N}{2}a_2$$

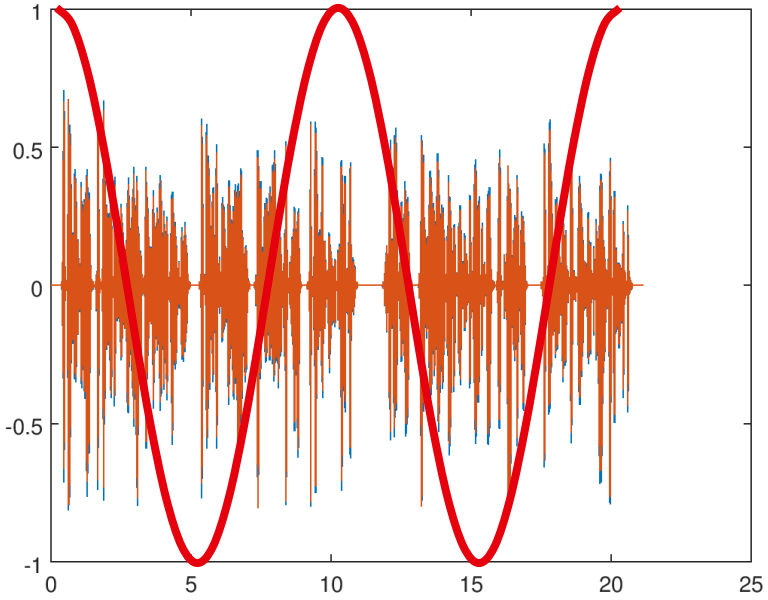
が残る.

たとえば

$$\cos\left(\frac{2\pi}{N}\{i\}\cdot 2\right)$$

がどのくらい入っているかを調べたい

内積を取る ▶ 成分の抜出し



もともと，この総和は

$$\{x_i\} = \begin{Bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{Bmatrix}$$

なので，これとの内積でも

▶ 内積を計算

たとえば

$$\cos\left(\frac{2\pi}{N}\{i\}\cdot 2\right)$$

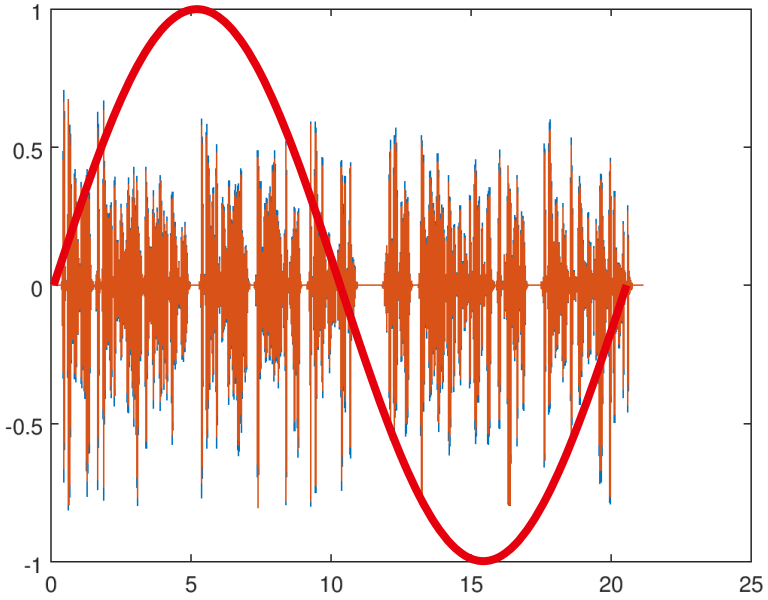
がどのくらい入っているかを調べたい

$$\frac{N}{2}a_2$$

が残る．

2倍波の内積でも対応する部分のみ残る

内積を取る ▶ 成分の抜出し



$$\frac{a_0}{2}\cos\left(\frac{2\pi}{N}\{i\}\cdot 0\right)$$
$$+a_1\cos\left(\frac{2\pi}{N}\{i\}\cdot 1\right)$$
$$+a_2\cos\left(\frac{2\pi}{N}\{i\}\cdot 2\right)$$
$$\vdots$$

$$+b_1\sin\left(\frac{2\pi}{N}\{i\}\cdot 1\right)$$
$$+b_2\sin\left(\frac{2\pi}{N}\{i\}\cdot 2\right)$$
$$\vdots$$

たとえば

$$\sin\left(\frac{2\pi}{N}\{i\}\cdot 1\right)$$

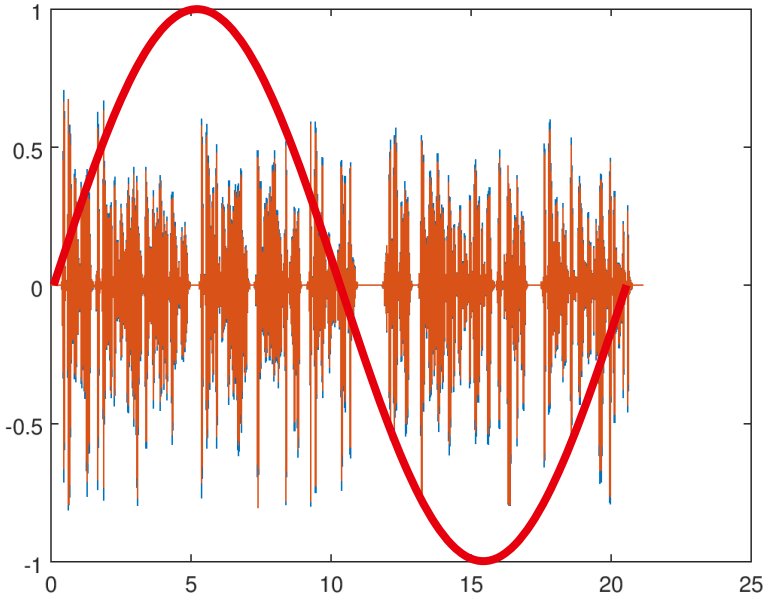
がどのくらい入っているかを調べたい

▶ 内積を計算

$$\sin\alpha\cos\beta=\frac{1}{2}[\sin(\alpha+\beta)+\sin(\alpha-\beta)]$$
$$\sin\alpha\sin\beta=\frac{1}{2}[\cos(\alpha+\beta)-\cos(\alpha-\beta)]$$
$$\cos\alpha\cos\beta=\frac{1}{2}[\cos(\alpha+\beta)+\cos(\alpha-\beta)]$$

中身が一致しない場合は波のまま▸総和を取れば0

内積を取る ▶ 成分の抜出し



$$\frac{a_0}{2}\cos\left(\frac{2\pi}{N}\{i\}\cdot 0\right)$$
$$+a_1\cos\left(\frac{2\pi}{N}\{i\}\cdot 1\right)$$
$$+a_2\cos\left(\frac{2\pi}{N}\{i\}\cdot 2\right)$$
$$\vdots$$

$$+b_1\sin\left(\frac{2\pi}{N}\{i\}\cdot 1\right)$$
$$+b_2\sin\left(\frac{2\pi}{N}\{i\}\cdot 2\right)$$
$$\vdots$$

▶ 内積を計算

$$\frac{N}{2}b_1$$

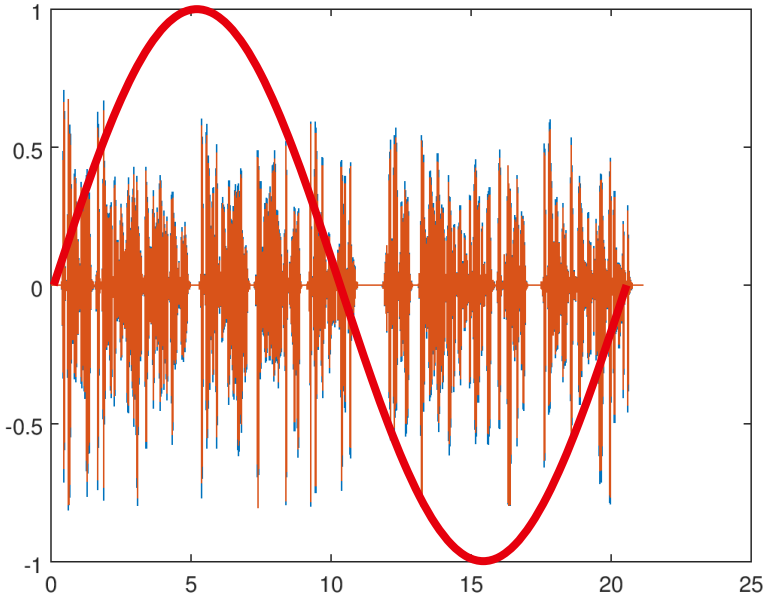
が残る.

たとえば

$$\sin\left(\frac{2\pi}{N}\{i\}\cdot 1\right)$$

がどのくらい入っているかを調べたい

内積を取る ▶ 成分の抜出し



もともと，この総和は

$$\{x_i\} = \begin{Bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{Bmatrix}$$

なので，これとの内積でも

▶ 内積を計算

たとえば

$$\sin\left(\frac{2\pi}{N}\{i\}\cdot 1\right)$$

がどのくらい入っているかを調べたい

$$\frac{N}{2}b_1$$

が残る．

sin と cos についても一致した場合のみ残る

総当たりで内積を取る ▶ 分解できる

$$\begin{bmatrix} \cos\left(\frac{2\pi}{N}\{i\}^T \cdot 0\right) \\ \cos\left(\frac{2\pi}{N}\{i\}^T \cdot 1\right) \\ \vdots \\ \cos\left(\frac{2\pi}{N}\{i\}^T \cdot \left(\frac{N}{2}\right)\right) \\ \sin\left(\frac{2\pi}{N}\{i\}^T \cdot 1\right) \\ \vdots \\ \sin\left(\frac{2\pi}{N}\{i\}^T \cdot \left(\frac{N}{2}-1\right)\right) \end{bmatrix}$$

自分で用意するので既知情報

※N は偶数とする.

$$\{x_i\} = \frac{N}{2}\begin{Bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{\frac{N}{2}} \\ b_1 \\ \vdots \\ b_{\frac{N}{2}-1} \end{Bmatrix}$$

計測データ

ほしい値

総当たりで内積を取る ▶ 分解できる

ただし，通常はデータの扱いの都合で複素数を用いる.

$$\begin{bmatrix} \exp\left(-\mathrm{i}\frac{2\pi}{N}\{i\}^T\cdot 0\right) \\ \exp\left(-\mathrm{i}\frac{2\pi}{N}\{i\}^T\cdot 1\right) \\ \vdots \\ \exp\left(-\mathrm{i}\frac{2\pi}{N}\{i\}^T\cdot (N-1)\right) \end{bmatrix} \{x_i\} = \frac{N}{2}\begin{Bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{N-1} \end{Bmatrix}$$

計測データ

ほしい値

ここで i（立体）は虚数単位，$\{i\}$ は ゼロからの整数列であることに注意.
内積は共役転置して掛合せるため符号が反転している.

$$\left[\exp\left(-\mathrm{i}\frac{2\pi}{N}\{i\}\{i\}^T\right)\right]\{x_i\} = \frac{N}{2}\{c_i\}$$

TOHOKU UNIVERSITY

高速フーリエ変換（FFT）

離散フーリエ変換は原理上できるが...

プログラムで書くと

```
>> U = exp(i*2*pi*[0:N-1]'*[0:N-1]/N);
>> X = U'*x;
```

となるのだが，おそらく通常のPCでは U のサイズ

$211289 \times 211289 \simeq 446$ 億

には対応できない.

そこで高効率に離散フーリエ変換を計算する

高速フーリエ変換（FFT）

を用いる

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 ◀変換前と同じ要素数
>> X(50000) ◀例えば50000番のデータは
ans = -5.5782 + 3.2176i ◀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
```

高速フーリエ変換について簡略化すると．．．

対称性を利用して効率的に係数を計算する方法である

$W_N = \exp\left(-\mathrm{i}\frac{2\pi}{N}\right)$ として，$N = 8$の場合を例として考える

▲以下，下付き8を省略

$$\begin{Bmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \\ V_6 \\ V_7 \end{Bmatrix} = \begin{pmatrix} W^0 & W^0 & W^0 & W^0 & W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 & W^4 & W^5 & W^6 & W^7 \\ W^0 & W^2 & W^4 & W^6 & W^8 & W^{10} & W^{12} & W^{14} \\ W^0 & W^3 & W^6 & W^9 & W^{12} & W^{15} & W^{18} & W^{21} \\ W^0 & W^4 & W^8 & W^{12} & W^{16} & W^{20} & W^{24} & W^{28} \\ W^0 & W^5 & W^{10} & W^{15} & W^{20} & W^{25} & W^{30} & W^{35} \\ W^0 & W^6 & W^{12} & W^{18} & W^{24} & W^{30} & W^{36} & W^{42} \\ W^0 & W^7 & W^{14} & W^{21} & W^{28} & W^{35} & W^{42} & W^{49} \end{pmatrix} \begin{Bmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{Bmatrix}$$

8回の掛け算

×8行

この行列を分解すると．．．

高速フーリエ変換

$$\begin{Bmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \\ V_6 \\ V_7 \end{Bmatrix} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & W^1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & W^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & W^3 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -W^1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -W^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -W^3 \end{pmatrix} \begin{pmatrix} I_2 & I_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & I_2 & W^2 I_2 \\ I_2 & -I_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & I_2 & -W^2 I_2 \end{pmatrix} \begin{pmatrix} I_4 & I_4 \\ I_4 & -I_4 \end{pmatrix} \begin{Bmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{Bmatrix}$$

変形は
各自追って

2回の掛け算
×8行

2回の掛け算
×8行

2回の掛け算
×8行

8×8＝64 回 ▶ 3×2×8＝48 回に減少

高速フーリエ変換

一般に,

$2^n \times 2^n$行列 ▶ 分解 ▶ n個の$2^n \times 2^n$行列

（ざっくりした）
掛け算の回数

※厳密には1倍や-1倍，i倍は計算負荷が軽いのだが概算で

2^n回 × 2^n行

n個の行列 × 2回 × 2^n行

計算量

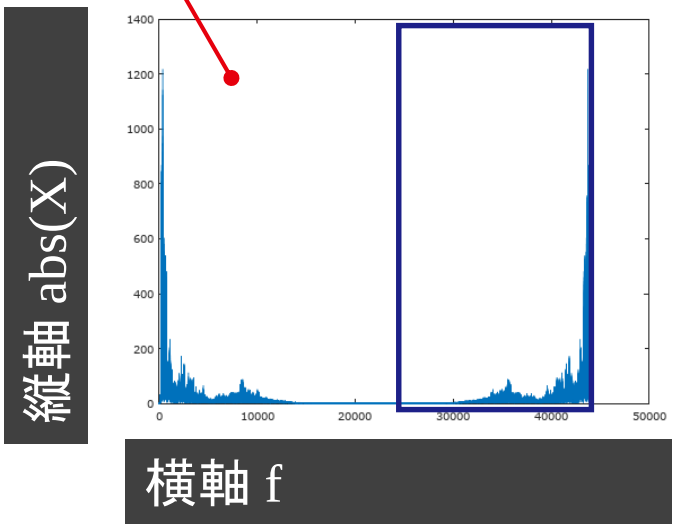
$O(N^2)$ ▶ $O(N \log N)$ に減小!!

例えば1024個のデータなら，100万回程度の計算を，1万回以下にする

FFT後の横軸(周波数軸)とFFTシフト

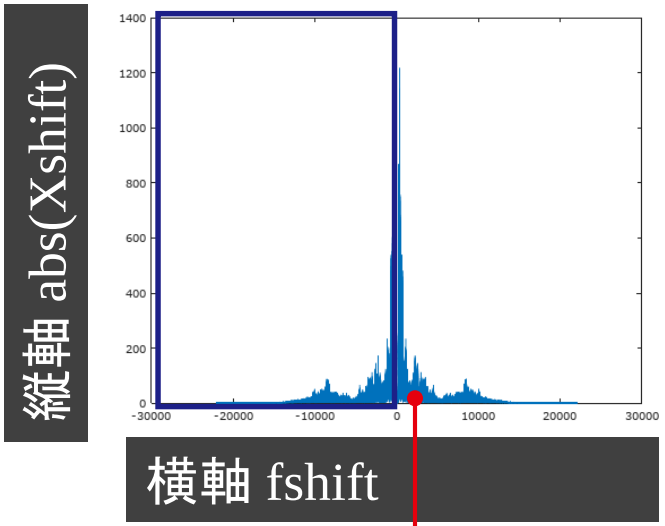

```
>> f=linspace(0,freq-freq/N,N);
>> plot(f,abs(X))
```

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

TOHOKU UNIVERSITY

FFTを使った応用

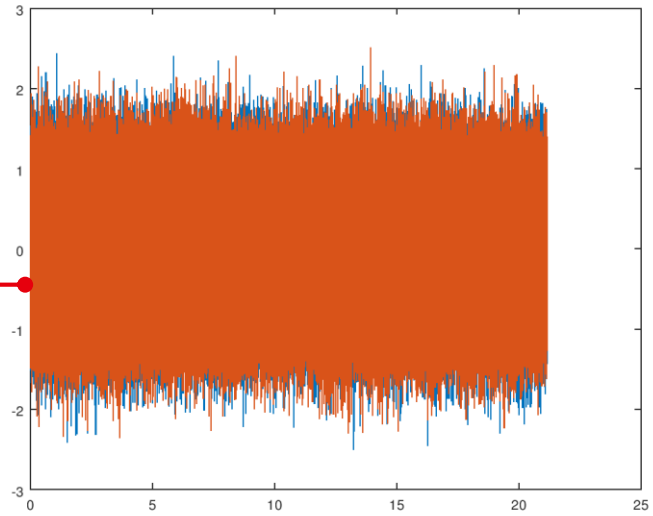
ノイズ除去 (1/3)

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));
```

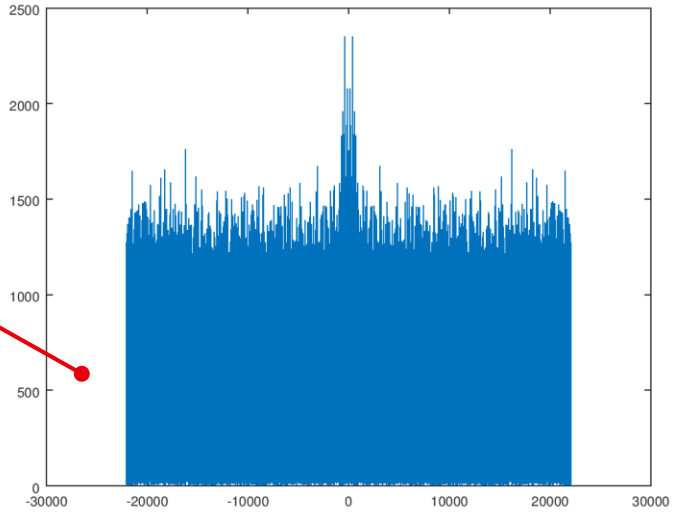
↳ ノイズ入り信号x_n: xと同サイズの乱数を作り重畳

```
>> 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)))
```



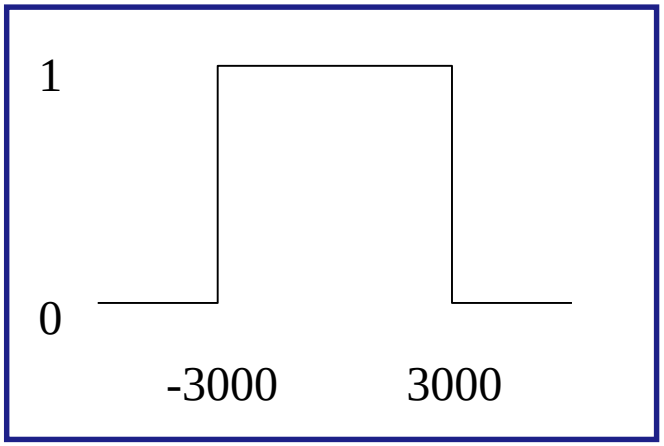
ノイズ除去 (2/3)

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

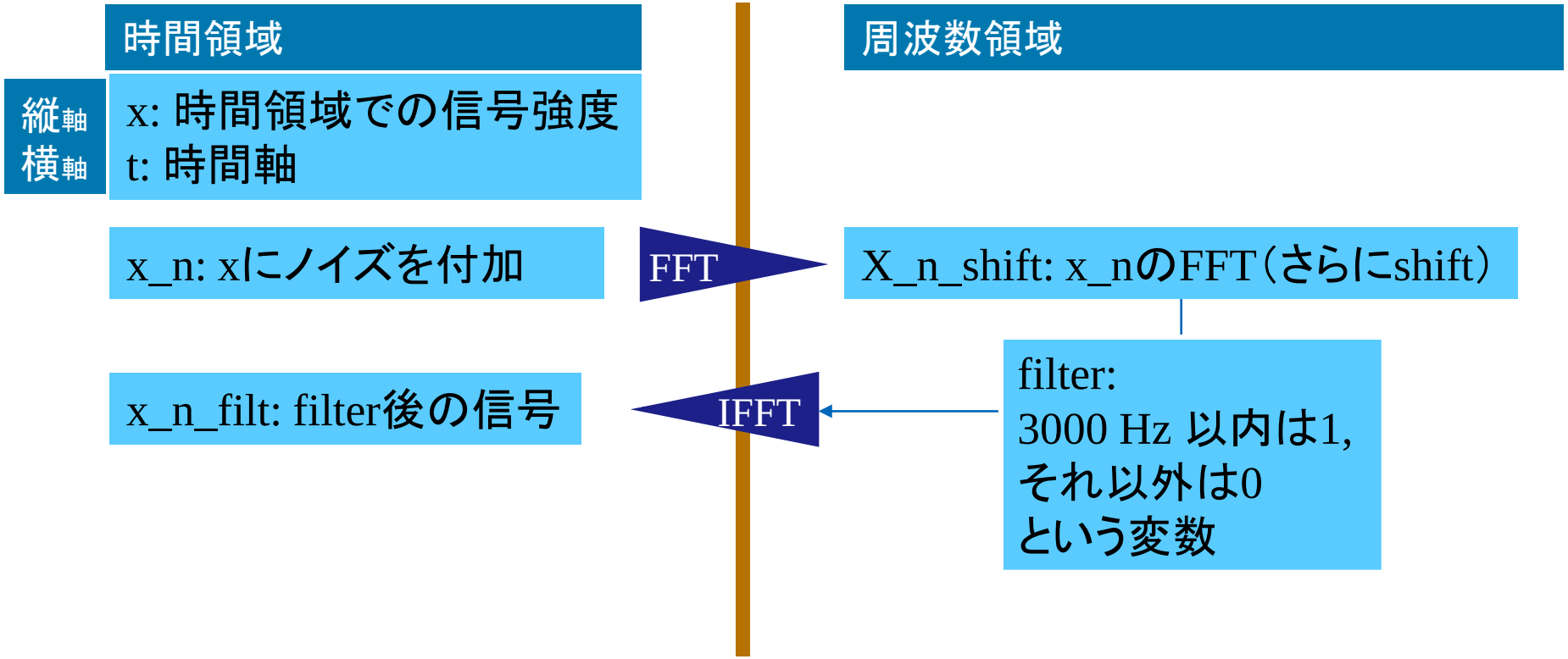
- fftshiftのシフトを元に戻す．

```
>> 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'));
```



課題 7

1. test.wav をFFTにより周波数分解し，元の信号（時間-強度）と分解後の信号（周波数-強度）のグラフを並べて表示せよ.
2. 1.で分解した信号について，「強度の高い」信号（閾値は，最強の信号の1/100まで，上位1000番まで，など任意に設定し，コメントアウトに記入すること）を残して強度の低い信号をカットし，逆FFTによって音声を復元せよ.（信号の圧縮）
3. test_w_noise.wavファイルはtest.wavに「ある周波数の純音」をノイズとして重畳して作成したファイルである. このtest_w_noise.wavからノイズを取り除くプログラムを作成せよ.（完全には無理なのでできる限り. ）

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)
```

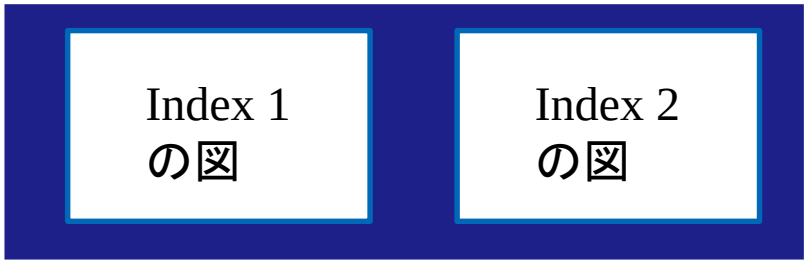
というコマンドが使える.

▲ 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が作業ディレクトリに
存在することを前提として作成しtest.wavを添付しないように.