認識行動システム論

第7回

梶本裕之

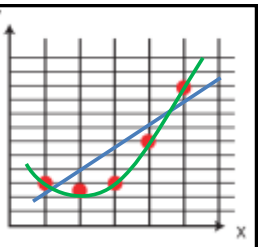
日程（多少変更）

12/03 信号処理と行列
12/10 中間テスト(授業時間中)
12/17 信号処理（アナログ・デジタル）
01/07 古典制御の基礎
01/14 ロボティクス
01/21 画像処理
01/28 期末テスト(授業時間中)

次回は中間テスト

前回のレポート課題(1)

次のデータ系列に対して，
Scilabを用いて，
(1) 直線による近似，
(2) 2次曲線による近似を適用，
パラメータを求め，
曲線とデータをグラフに描け



x	1.0	2.0	3.0	4.0	5.0
y	3.0	2.5	3.0	6.0	10.0

なおScilabでは行列Aの擬似逆行列はpinv(A)で直接求めることができる．
当然自分でinv(A'*A)*A'とやっても同じ．

(1-1) 直線による近似

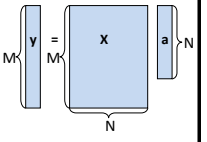
$$y = a_1 x + a_2$$

これは

$$y = a_1 x_1 + a_2 x_2 \quad \text{where} \quad x_2 = 1$$

とみなせる．

$$\begin{aligned} y_1 &= a_1 x_{11} + a_2 1 \\ y_2 &= a_1 x_{21} + a_2 1 \\ &\vdots \\ y_M &= a_1 x_{M1} + a_2 1 \end{aligned} \qquad \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{bmatrix} = \begin{bmatrix} x_{11} & 1 \\ x_{21} & 1 \\ \vdots & \vdots \\ x_{M1} & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$$



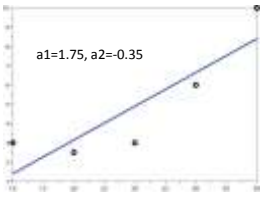
よって，

$$\mathbf{a} = \mathbf{X}^{\#}\mathbf{y} \quad \text{where} \quad \mathbf{X}^{\#} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}$$

により二つの未知パラメータを求めることが出来る．

(1-1) 直線による近似

$$\begin{bmatrix} 3.0 \\ 2.5 \\ 3.0 \\ 6.0 \\ 10.0 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \\ 4 & 1 \\ 5 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$$



Scilabコード例

```
x=[1;2;3;4;5];
y=[3.0; 2.5; 3.0; 6.0; 10.0];
plot2d(x,y,style=[-9] );//元のデータをプロット

X=[1,1;2,1;3,1;4,1;5,1];        //擬似逆行列のための行列
a=inv(X'*X)*X'*y                //係数を最小二乗法で求める.
y_fitting = a(1) * x + a(2);    //フィッティング結果の直線
plot(x,y_fitting);              //表示
```

(1-2) 2次曲線による近似

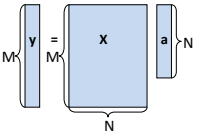
$$y = a_1 x^2 + a_2 x + a_3$$

これは

$$y = a_1 x_1 + a_2 x_2 + a_3 x_3$$

とみなせる．

$$\begin{aligned} y_1 &= a_1 x_{11} + a_2 x_{12} + a_2 1 \\ y_2 &= a_1 x_{21} + a_2 x_{22} + a_2 1 \\ &\vdots \\ y_M &= a_1 x_{M1} + a_2 x_{M2} + a_2 1 \end{aligned}$$



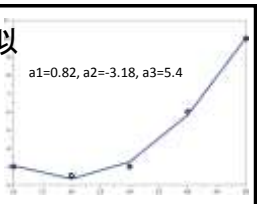
よって，

$$\mathbf{a} = \mathbf{X}^{\#}\mathbf{y} \quad \text{where} \quad \mathbf{X}^{\#} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}$$

により二つの未知パラメータを求めることが出来る．

(1-2)2次曲線による近似

$$\begin{bmatrix} 3.0 \\ 2.5 \\ 3.0 \\ 6.0 \\ 10.0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 4 & 2 & 1 \\ 9 & 3 & 1 \\ 16 & 4 & 1 \\ 25 & 5 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

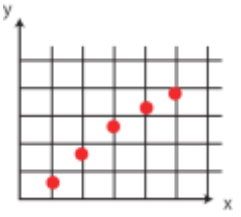


Scilabコード例

```
x=[1;2;3;4;5];
y=[3.0; 2.5; 3.0; 6.0; 10.0];
plot2d(x,y,style=[-9] );                    //元のデータをプロット

X=[1,1,1;4,2,1;9,3,1;16,4,1;25,5,1];   //擬似逆行列のための行列
a=inv(X'*X)*X'*y                            //係数を最小二乗法で求める.
y_fitting = a(1).*x .*x + a(2)*x +a(3); //フィッティング結果を示す
plot(x,y_fitting);                          //表示
```

前回のレポート課題(2)

次のデータ系列に対して，
y=a1 * log(x) + a2
を仮定してパラメータを求め，
曲線とデータをグラフに描け
（やや難？）



x	1.0	2.0	3.0	4.0	5.0
y	0.5	1.9	2.7	3.3	3.7

8

(2)logによる近似

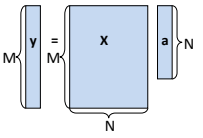
$$y = a_1 \log(x) + a_2$$

これは

$$y = a_1 x_1 + a_2 x_2 \quad \text{where} \quad x_2 = 1$$

とみなせる.

$$\begin{matrix} y_1 = a_1 \log(x_{11}) + a_2 1 \\ y_2 = a_1 \log(x_{21}) + a_2 1 \\ \vdots \\ y_M = a_1 \log(x_{M1}) + a_2 1 \end{matrix} \quad \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{bmatrix} = \begin{bmatrix} x_{11} & 1 \\ x_{21} & 1 \\ \vdots & \vdots \\ x_{M1} & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$$



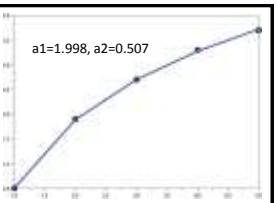
よって,

$$\mathbf{a} = \mathbf{X}^{\#}\mathbf{y} \quad \text{where} \quad \mathbf{X}^{\#} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}$$

により二つの未知パラメータを求めることが出来る.

(2)logによる近似

$$\begin{bmatrix} 0.5 \\ 1.9 \\ 2.7 \\ 3.3 \\ 3.7 \end{bmatrix} = \begin{bmatrix} \log(1) & 1 \\ \log(2) & 1 \\ \log(3) & 1 \\ \log(4) & 1 \\ \log(5) & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$$



Scilabコード例

```
x=[1;2;3;4;5];
y=[0.5; 1.9; 2.7; 3.3; 3.7];
plot2d(x,y,style=[-9] );                    //元のデータをプロット

//擬似逆行列のための行列
X=[log(1),1; log(2),1; log(3),1; log(4),1;log(5),1];
a=inv(X'*X)*X'*y                            //係数を最小二乗法で求める.
y_fitting = a(1)*log(x)+ a(2);              //フィッティング結果を示す
plot(x,y_fitting);                          //表示
```

10

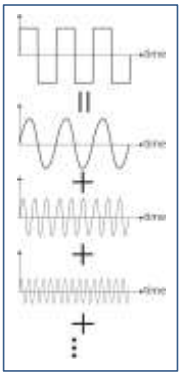
レポートの成績評価とは関係なく，
これらの課題は，一度は打ち込んで
結果を見ることを強く勧めます.

最小二乗法は信号処理，制御，統計
などあらゆる分野で使われています.

11

信号処理の基礎

（復習）:フーリエ級数展開



周期Tの波形 f(t)は次のように分解できる

$$f(t) = a_0 + \sum_{m=1}^{\infty} a_m \cos(2\pi mt/T) + \sum_{m=1}^{\infty} b_m \sin(2\pi mt/T)$$

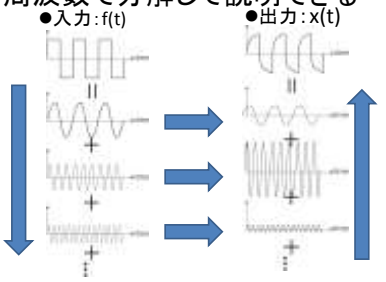
$$a_0 = \frac{1}{T}\int_0^T f(t)dt$$ 平均値（DC成分）

$$a_m = \frac{2}{T}\int_0^T f(t)\cos(2\pi mt/T)dt$$

$$b_m = \frac{2}{T}\int_0^T f(t)\sin(2\pi mt/T)dt$$

※この授業では係数は気にしない.

（復習:フーリエ級数展開）
歪みを周波数で分解して説明できる



（1）入力f(t)を周波数分解する
（2）周波数ごとの出力の振幅と位相を求める.
（3）合計すると出力が得られる.
これを連続関数で考えるとどうなるか？

（復習）フーリエ変換

フーリエ級数展開は周期的な信号を分解するのに使われた.
フーリエ変換は周期的ではない信号に対する変換.
Tを無限大とした極限から導かれる.
逆フーリエ変換によって元に戻すことができる.

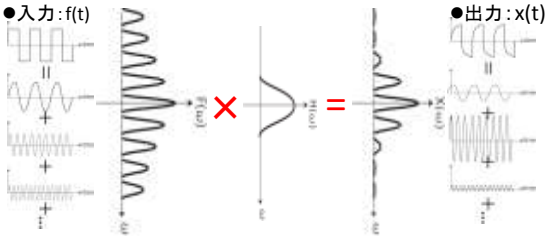
フーリエ変換

$$F(\omega) = \int_{-\infty}^{\infty} f(t)\exp(-j\omega t)dt$$

逆フーリエ変換

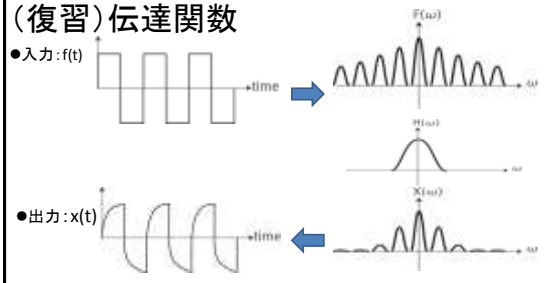
$$f(t) = \int_{-\infty}^{\infty} F(\omega)\exp(j\omega t)d\omega$$

（復習）入出力の関係:関数同士の掛け算



（1）入力f(t)を周波数分解⇒F(ω)
（2）周波数ごとにどれだけ振幅と位相が変わるか:H(ω)
（3）出力（のフーリエ変換）:X(ω)=H(ω)*F(ω)
（4）逆フーリエ変換すると出力が得られる:x(t)

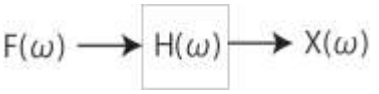
（復習）伝達関数



フーリエ空間では、入出力は単なる掛け算で表される.
X(ω)＝H(ω)×F(ω)

この入出力関係を定義するシステムの性質H(ω)を
伝達関数と呼ぶ.

（復習）伝達関数(Transfer Function)

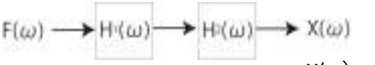


X(ω)＝H(ω)×F(ω)

システムの入出力関係.「伝わり方」を周波数空間で記述したもの.

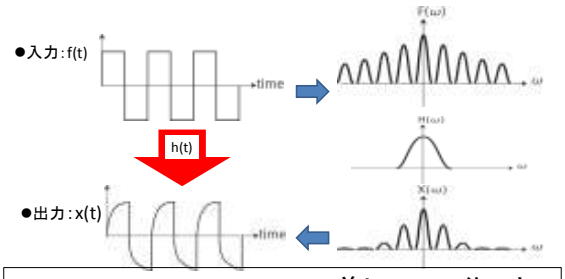
伝達関数は単純な掛算でよい
複数のフィルタの特性は周波数領域では単純に掛け算をすればよい

（例）
ローパスフィルタとハイパスフィルタを続けてかけることでバンドパスフィルタを作成可能
フィルタをかける順番にも依存しない.



X(ω)＝H1(ω)×H2(ω)×F(ω)

周波数領域ではなく，時間領域のまま議論できないか？



X(ω)＝H(ω)×F(ω)：周波数領域で美しいのは分った．
時間的な現象として何が起きているのか分からない．

式で考えよう

X(ω)＝H(ω)×F(ω)

フーリエ変換
$$F(\omega)=\int_{-\infty}^{\infty} f(t)\exp(-j\omega t)dt$$
逆フーリエ変換
$$f(t)=\int_{-\infty}^{\infty} F(\omega)\exp(j\omega t)d\omega$$

両辺を逆フーリエ変換すれば時間領域の信号に戻る．

$$x(t)=$$
$$=$$
$$=$$
$$=$$

$$\int_{-\infty}^{\infty} H(\omega)\exp(j\omega(t-\tau))d\omega=\int_{-\infty}^{\infty}(H(\omega)\exp(-j\omega\tau))\exp(j\omega t)d\omega$$
$$\int_{-\infty}^{\infty} h(t-\tau)\exp(-j\omega t))dt=\int_{-\infty}^{\infty} h(t')\exp(-j\omega(t'+\tau))dt'$$
$$=\exp(-j\omega\tau)\int_{-\infty}^{\infty} h(t')\exp(-j\omega t'))dt'$$
$$=H(\omega)\exp(-j\omega\tau)$$

逆順の計算もしておく（ふつうはこちら）

$$x(t)=\int_{-\infty}^{\infty} f(\tau)h(t-\tau)d\tau$$

フーリエ変換
$$F(\omega)=\int_{-\infty}^{\infty} f(t)\exp(-j\omega t)dt$$
逆フーリエ変換
$$f(t)=\int_{-\infty}^{\infty} F(\omega)\exp(j\omega t)d\omega$$

両辺をフーリエ変換．

$$X(\omega)=\int_{-\infty}^{\infty}\exp(-j\omega t)dt\int_{-\infty}^{\infty} f(\tau)h(t-\tau)d\tau$$
$$=\int_{-\infty}^{\infty}\exp(-j\omega(\tau+(t-\tau))dt\int_{-\infty}^{\infty} f(\tau)h(t-\tau)d\tau$$
$$=\int_{-\infty}^{\infty}\exp(-j\omega\tau)\cdot\exp(-j\omega(t-\tau))dt\int_{-\infty}^{\infty} f(\tau)h(t-\tau)d\tau$$
$$=\int_{-\infty}^{\infty}\quad d\tau\int_{-\infty}^{\infty}\quad dt$$
$$=\int_{-\infty}^{\infty}\quad d\tau\int_{-\infty}^{\infty}\quad dt'$$
$$=F(\omega)H(\omega)$$

コンボリューション定理

$$X(\omega)=F(\omega)H(\omega)=H(\omega)F(\omega)$$

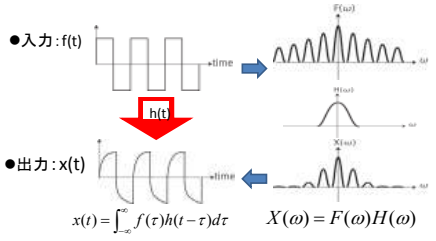
フーリエ逆変換　　フーリエ変換

$$x(t)=\int_{-\infty}^{\infty} f(\tau)h(t-\tau)d\tau=\int_{-\infty}^{\infty} h(\tau)f(t-\tau)d\tau$$

簡略化のため次のようにも表記される

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

コンボリューション定理の意味するところ（1）

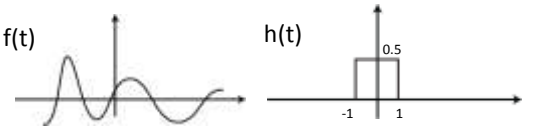


$$x(t)=\int_{-\infty}^{\infty} f(\tau)h(t-\tau)d\tau \qquad X(\omega)=F(\omega)H(\omega)$$

- h(t)のフーリエ変換がH(ω)であるとする．
- 周波数領域でフィルタH(ω)をかけることは，
時間領域では，入力信号x(t)に対する関数h(t)の畳み込み積分（コンボリューション）として表現される．

コンボリューション定理の意味するところ（2）

$$x(t)=\int_{-\infty}^{\infty} h(\tau)f(t-\tau)d\tau$$

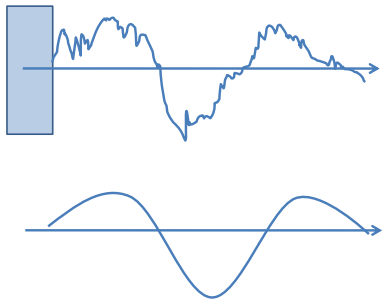


例えば，h(t)=0.5　(-1<t<1)なら，

$$x(t)=$$

これは，f(t)を平均化していくフィルタ

平均化？



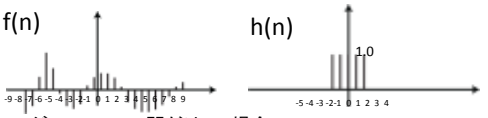
ノイズを「ならし」て大域的な特徴をつかむ

離散化による理解

$$x(t)=\int_{-\infty}^{\infty}h(\tau)f(t-\tau)d\tau \Rightarrow x(n)=\sum_{i=-\infty}^{\infty}h(i)f(n-i)$$

x(n)=...+ h(-4)f(n+4) + h(-3)f(n+3) +...+ h(3)f(n-3) + h(4)f(n-4)+...

f(n)　　h(n)



h(n)が，n=-2～2の間だけ1の場合，

x(1)=f(3) + f(2) + f(1) + f(0) + f(-1)
x(2)=f(4) + f(3) + f(2) + f(1) + f(0)
x(3)=f(5) + f(4) + f(3) + f(2) + f(1)
x(4)=f(6) + f(5) + f(4) + f(3) + f(2)
x(n)=f(n+2) + f(n+1) + f(n) + f(n-1) + f(n-2)

出力xは，入力fの「平均化」になっている．

（復習）フーリエ変換の計算例：矩形波

$$h(t)=\begin{cases}1/2 & -1\le t\le 1\\ 0 & otherwise\end{cases}$$

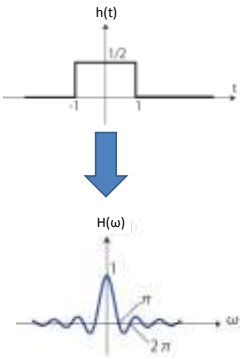
$$H(\omega)=\int_{t=-\infty}^{\infty}f(t)\exp(-j\omega t)dt$$

$$=\left[\frac{1}{-j2\omega}\exp(-j\omega t)\right]_{-1}^{1}$$

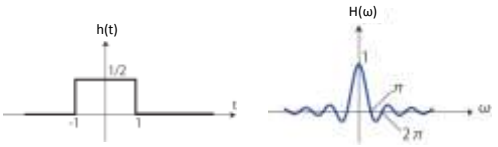
$$=\frac{1}{-j2\omega}(\exp(-j\omega)-\exp(j\omega))$$

$$=\frac{1}{-j2\omega}(\cos(\omega)-j\sin(\omega)-\cos(\omega)-j\sin(\omega))$$

$$=\frac{-j\sin(\omega)}{-j\omega}$$



h(t)とH(ω)の関係：フーリエ変換



つまり，h(t)のフーリエ変換H(ω)は，
低い周波数で大きな値をとり，高い周波数で小さな値をとる．

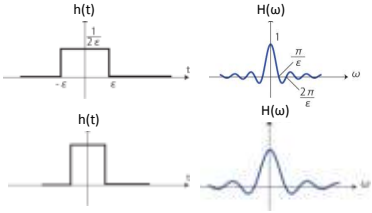
X(ω)=H(ω) × F(ω)

を思い出すと，H(ω)は，
「X(ω)の成分のうち，低い周波数を通過させ，高い周波数を阻止する」
すなわち，低域通過フィルタ（LPF：Low Pass Filter）である．

時間領域での「平均化（平滑化）フィルタ」
≒周波数領域での「ローパスフィルタ」

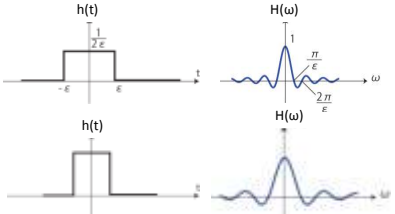
（復習）矩形波の幅が変わると？

$$h(t)=\begin{cases}\dfrac{1}{2\varepsilon} & -\varepsilon\le t\le\varepsilon\\ 0 & otherwise\end{cases} \Rightarrow H(\omega)=$$



矩形波の幅を狭くする ⇒ フーリエ変換結果は幅広に

平均化の時間幅と周波数帯域の関係



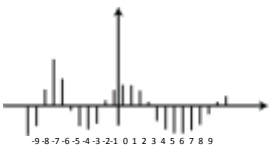
矩形波の幅を狭くする ⇒ フーリエ変換結果は幅広に

時間的な平均化（平滑化）フィルタの幅が広いほど
周波数的には低い周波数しか通さなくなる．

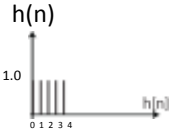
時間軸の離散化：FIRフィルタによる実装

$$x(n)=\sum_{i=0}^{\infty}h(i)f(n-i)$$

f(n)



h(n)



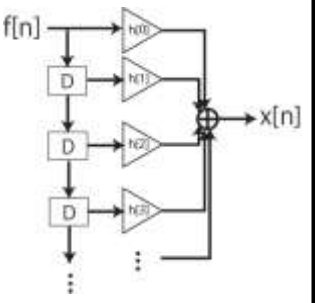
i=0から始める：未来のデータ が使えないことを意味する．
この例は，元データf(n)を，４個平均して出力する．

- 未来のデータが使えない例：リアルタイム制御
- 先のデータが使える例：画像処理

FIRフィルタの図的理解

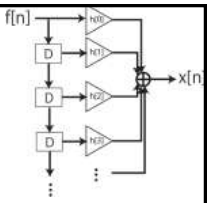
$x(n)=$

D:Delay，遅延器，メモリ．
h[n]：増幅器



FIR＝Finite Impulse Response
個々のインパルス応答を有限個足し合わせたもの．

（参考）FIRフィルタと行列



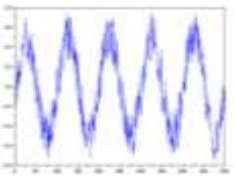
$$x(n)=\sum_{i=0}^{\infty}h(i)f(n-i)$$

フィルタが3つのメモリを持つ場合

$$x(n)=h(0)f(n)+h(1)f(n-1)+h(2)f(n-2)$$

$$\begin{aligned}
x(0)&=h(0)f(0)\\
x(1)&=h(0)f(1)+h(1)f(0)\\
x(2)&=h(0)f(2)+h(1)f(1)+h(2)f(0)\\
x(3)&=h(0)f(3)+h(1)f(2)+h(2)f(1)\\
x(4)&=h(0)f(4)+h(1)f(3)+h(2)f(2)\\
x(5)&=h(0)f(5)+h(1)f(4)+h(2)f(3)
\end{aligned}
\quad
\begin{bmatrix}x(0)\\x(1)\\x(2)\\x(3)\\x(4)\end{bmatrix}=\begin{bmatrix}h(0)&0&0&0&0\\h(1)&h(0)&0&0&0\\h(2)&h(1)&h(0)&0&0\\0&h(2)&h(1)&h(0)&0\\0&0&h(2)&h(1)&h(0)\end{bmatrix}\begin{bmatrix}f(0)\\f(1)\\f(2)\\f(3)\\f(4)\end{bmatrix}$$

平滑化フィルタの実例（1）



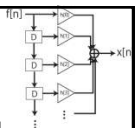
元の信号に
高周波ノイズが含まれている．

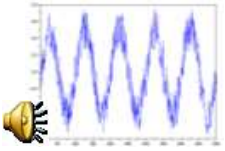
Scilabコード例

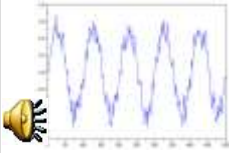
```
time = [0:0.01:100];

//振幅0.5の正弦波に最大振幅0.5のノイズが混入
wave=0.5*sin(time*2*%pi) + 0.5*(rand(time)-0.5);
playsnd(wave);
savewave('wave.wav',wave);
plot(wave(1:500));
```

平滑化フィルタの実例（2）

メモリを三つ持ったFIRフィルタによって平滑化





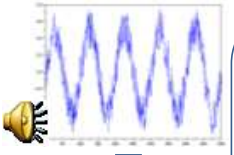


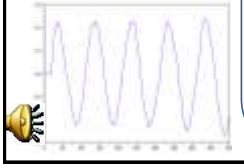
Scilabコード例

```
time = [0:0.01:100];
//振幅0.5の正弦波に最大振幅0.5のノイズが混入した信号
wave=0.5*sin(time*2*%pi) + 0.5*(rand(time)-0.5);

out=zeros(wave);
//3つを平均する.
for n=3:length(wave),
 for i=0:2,
  out(n)=out(n)+wave(n-i)/3;
 end
end
playsnd(out);
savewave('wave.wav',out);
plot(out(1:500));
```

平滑化フィルタの実例（3）

メモリを20個持ったFIRフィルタによって平滑化

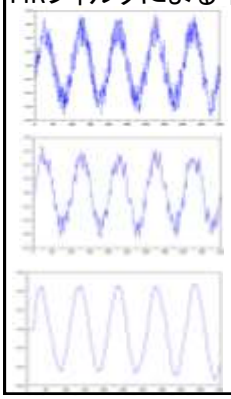


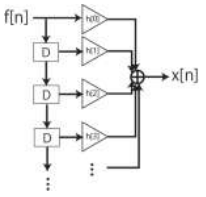


Scilabコード例

```
time = [0:0.01:100];
//振幅0.5の正弦波に最大振幅0.5のノイズが混入した信号
wave=0.5*sin(time*2*%pi) + 0.5*(rand(time)-0.5);

out=zeros(wave);

//20個を平均する.
for n=20:length(wave),
 for i=0:19,
  out(n)=out(n)+wave(n-i)/20;
 end
end

playsnd(out);
savewave('wave.wav',out);
plot(out(1:500));
```

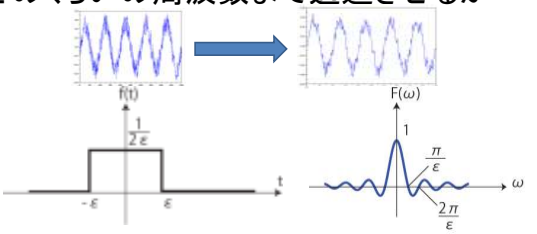
FIRフィルタによる平滑化の効果と弊害





ステップ数が多くなるほど
＜効果＞
平滑化の効果が高い
（＝低域の通過周波数が下がる）
＜弊害＞
計算量の増大
ステップ数分の「時間遅れ」が必ず生じる

どのくらいの周波数まで通過させるか

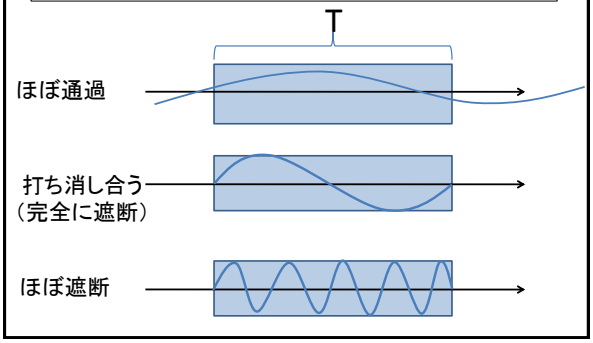


幅2εの矩形波のフーリエ変換：角周波数π/εで0

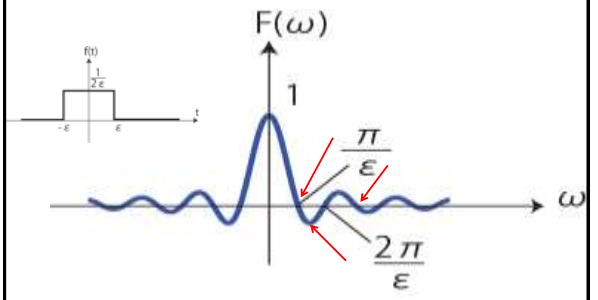
時間幅Tで平均化する場合：
角周波数2π/T（周波数1/T)(以上）の波を遮断.

平均化による遮断

時間幅Tの平均化：周波数1/T (以上）の波を遮断.



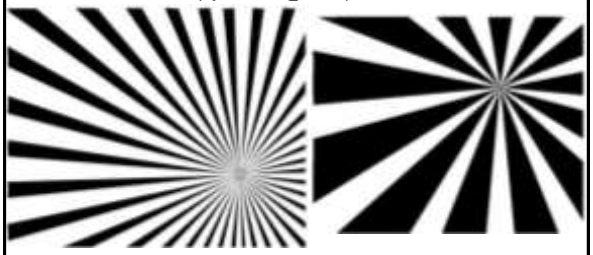
平均化によるローパス：完ぺきではない



特定の周波数は全く通さないが，高周波成分の遮断が周期的ふるまいを示す.

（参考）偽解像現象

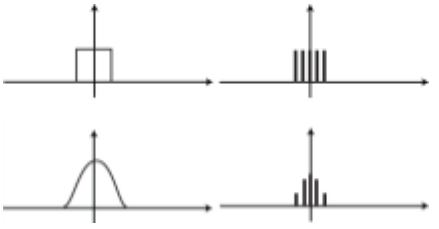
プロジェクタのピントをぼかす
人間のピントをぼかす



「ピンボケ」のローパスフィルタとしての特性が周期性をもつために生じる現象.

実際のローパス

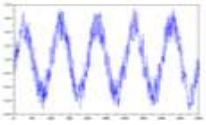
周波数上での周期的なふるまいを無くすため，なだらかにする.

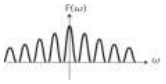


画像の世界では．．．「ガウシアン」

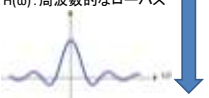
ローパスフィルタのまとめ

●入力：f(t)



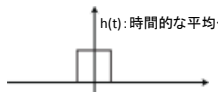


H(ω)：周波数的なローパス



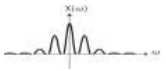
$X(\omega) = F(\omega)H(\omega)$

h(t)：時間的な平均化



$x(t) = \int_{-\infty}^{\infty} f(\tau)h(t-\tau)d\tau$

●出力：x(t)

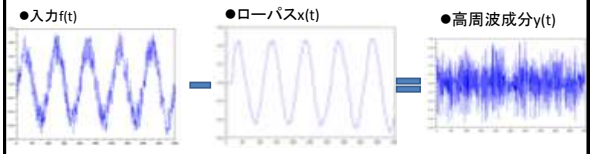


逆に高い周波数成分だけ取り出すには？

●ローパスフィルタ：低い周波数成分だけを取り出した

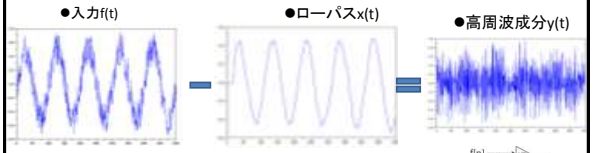
●元信号と低周波信号の差をとれば，高周波数成分だけ取り出せる？

●入力f(t)　●ローパスx(t)　●高周波成分y(t)

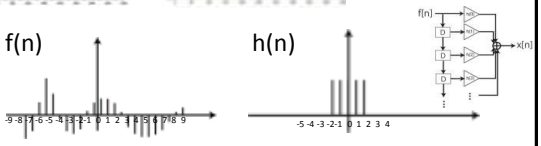


ハイパスフィルタのFIRフィルタによる実装

●入力f(t)　●ローパスx(t)　●高周波成分y(t)



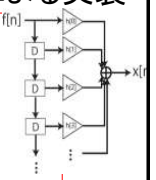
f(n)　h(n)



ローパス例：x(n)=(f(n+2) + f(n+1) + f(n) + f(n-1) + f(n-2))/5
ハイパス例：y(n)=f(n)-x(n)
=-1/5・f(n+2) -1/5・f(n+1) +4/5f(n) -1/5・f(n-1) -1/5・f(n-2)

ハイパスフィルタのFIRフィルタによる実装

ローパス：
強　x(n)=(f(n+2) + f(n+1) + f(n) + f(n-1) + f(n-2))/5
↑　x(n)=(f(n+1) + f(n) + f(n-1) + f(n-2))/4
↓　x(n)=(f(n+1) + f(n) + f(n-1))/3
弱　x(n)=(f(n) + f(n-1))/2



ハイパス：y(n)=f(n)-x(n)

ローパスが{強い・弱い}ほど，ハイパスは{弱く・強く}なる

最も簡単な場合：

$x(n)=(f(n) + f(n-1))/2$

ハイパス：

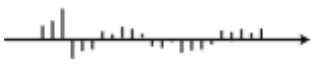
$y(n) = f(n)-x(n) =$

つまり，直前との「差分（微分）」.

ハイパスフィルタ≒微分フィルタ

●入力f(t)

●高周波成分y(t)



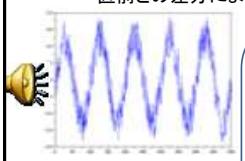
y(1) = (f(1) - f(0))/2
y(2) = (f(2) - f(1))/2
y(3) = (f(3) - f(2))/2…

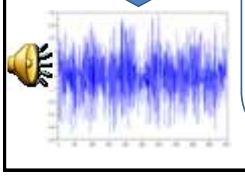
用語整理

（周波数表現）		（時間軸表現）
ローパスフィルタ	＝	低域通過フィルタ
ハイパスフィルタ	＝	高域通過フィルタ

ハイパスフィルタの例

直前との差分によってハイパス



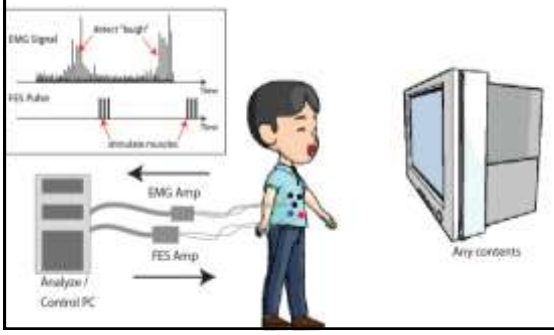


Scilabコード例

```
time = [0:0.01:100];
//振幅0.5の正弦波に最大振幅0.5のノイズが混入した信号
wave=0.5*sin(time*2*%pi) + 0.5*(rand(time)-0.5);

out=zeros(wave);

//差分をとる
for n=2:length(wave),
  out(n)=wave(n)-wave(n-1);
End

playsnd(out);
savewave('wave.wav',out);
plot(out(1:500));
```

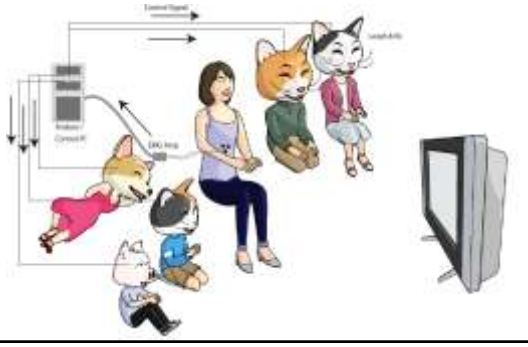
フィルタリング．．．研究の現場で

筋電計測による笑いの検出→増幅は可能か？



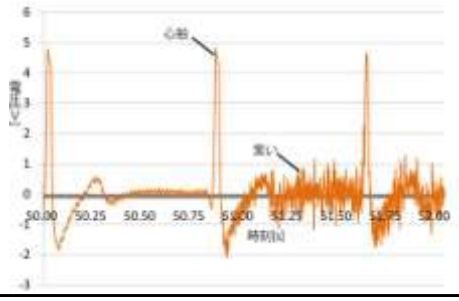
研究の現場で

筋電計測による笑いの検出→増幅は可能か？

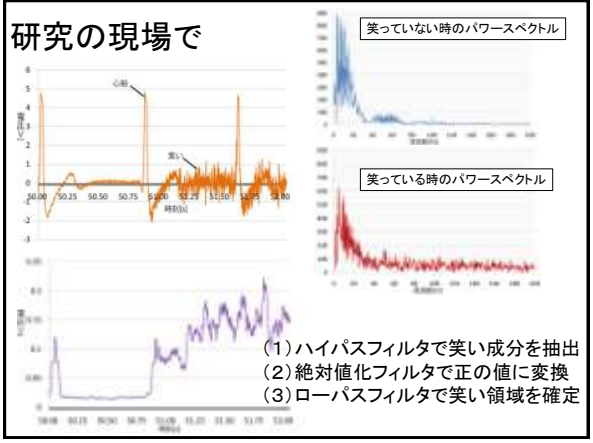


研究の現場で

筋電計測：
- ●心拍による成分：非常に大きいが，低周波
- ●笑いによる成分：小さいが，高周波



研究の現場で



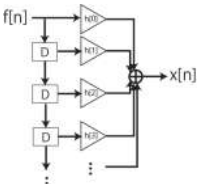
（1）ハイパスフィルタで笑い成分を抽出
（2）絶対値化フィルタで正の値に変換
（3）ローパスフィルタで笑い領域を確定

参考：エコー

エコー＝時間遅れ信号の重畳．
これもFIRフィルタで実装できる．

Scilabコード例

```
wave = loadwave('aiueo.wav');

out=zeros(wave);

//エコー（1000ステップ前の信号を重畳）
for n=1000:length(wave),
   out(n)=wave(n)+0.9*wave(n-999);
end

playsnd(out,11000);      //11kHzサンプリングで再生
savewave('wave.wav',out,[11000]);
```



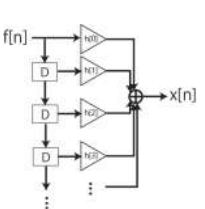

原音

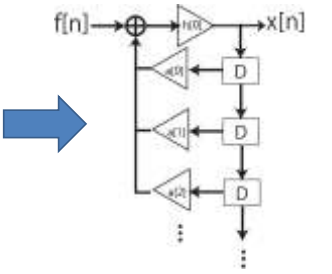

1000ステップ前の信号を重畳


1000ステップ前＋2000ステップ前の信号を重畳


沢山重畳

参考：IIRフィルタ



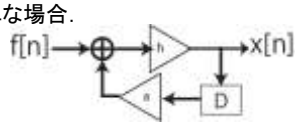


出力データを入力に帰還させると．．．

$x[n]=h[0]*(f[n]+a[0]*x[n-1]+a[1]*x[n-2]+...)$

IIRフィルタ(参考)

最も簡単な場合.



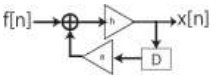
$$
\begin{aligned}
x[n] &= h(f[n]+ax[n-1]) \\
&= h(f[n]+ah(f[n-1]+ax[n-2])) \\
&= h(f[n]+ah(f[n-1]+ah(f[n]+ax[n-1]))) \\
&= h(f[n]+ahf[n-1]+a^2h^2f[n-2]+a^3h^3f[n-3]+\cdots) \\
&= h\sum_{i=0}^{\infty} a^i h^i f[n-i]
\end{aligned}
$$

メモリが一つにもかかわらず，無限の過去が影響する

IIR＝Infinite Impulse Response

IIRフィルタ(参考)

$$
\begin{aligned}
x[n] &= h(f[n]+ax[n-1]) \\
&= h(f[n]+ah(f[n-1]+ax[n-2])) \\
&= h(f[n]+ah(f[n-1]+ah(f[n]+ax[n-1]))) \\
&= h(f[n]+ahf[n-1]+a^2h^2f[n-2]+a^3h^3f[n-3]+\cdots) \\
&= h\sum_{i=0}^{\infty} a^i h^i f[n-i]
\end{aligned}
$$



メモリが一つにもかかわらず，無限の過去が影響する

＜利点＞
少ないメモリで等価的に大きな効果が得られる.
＜欠点＞
帰還ゲインaが大きいと発振する.
（式からah<1でなければならないのは明らか）

現在はメモリの制約が少ないためあまり使われない.

レポート課題

適当なwaveファイルに対して次の三つの操作を行う.
（1）FIRフィルタによるローパスフィルタをかけて音をくもらせる.
（2）FIRフィルタによる適当なハイパスフィルタをかけて音をとがらせる.
（3）エコーを掛けてカラオケのようにする.

Scilabのソースファイル，原音のwaveファイル，処理後のwaveファイルを添付すること.
（ただし添付ファイルは1Mbyte以下に抑えてください．）