インタラクティブシステム論
第5回

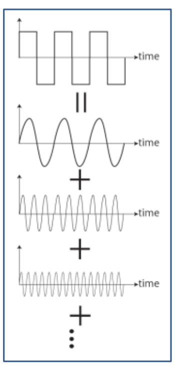
梶本裕之
Twitter ID kajimoto
ハッシュタグ #ninshiki

日程

4/10　イントロダクション
4/17 Scilabの紹介（西6号館3階PCルーム）
4/24　フーリエ変換
5/1 出張により休講
5/8　出張により休講⇒変更！
5/15フーリエ変換と線形システム
5/22　信号処理の基礎
5/29　出張により休講　⇒変更！
6/5　信号処理応用1（相関）
6/12　信号処理応用2（画像処理）中間確認レポート出題⇒変更！
6/19　ラプラス変換，
6/26　出張により休講
7/3　古典制御の基礎
7/10　行列
7/17　行列と最小二乗法
7/24　ロボティクス
8/2～8　期末テスト

休講が重なったため，中間確認テストを中間確認レポートに変更します（7/3講義前に提出）

（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）

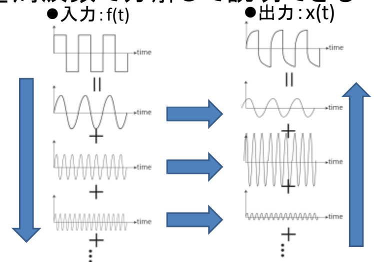
$$a_m = \frac{2}{T}\int_0^T f(t)\cos(2\pi mt/T)dt$$

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

※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

●入力：f(t)　●出力：x(t)



（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

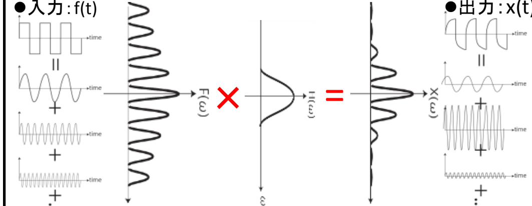
$$F(\omega) = \int_{-\infty}^{\infty} f(t)\exp(-j\omega t)dt$$

逆フーリエ変換

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

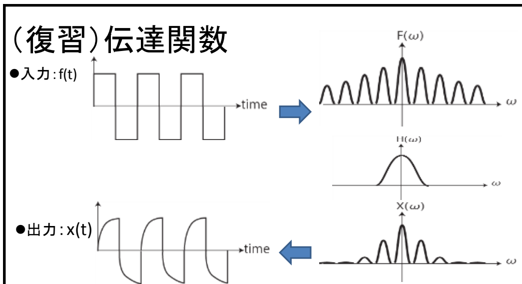
（復習）入出力の関係：関数同士の掛け算

●入力：f(t)　●出力：x(t)



（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)

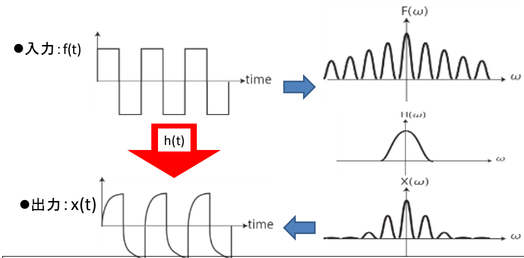
(復習)伝達関数



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．

今日の話題: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

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

X(ω)＝H(ω)×F(ω)

両辺を逆フーリエ変換すれば時間領域の信号に戻る．

$x(t)=$

$=$

$=$

$=$

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．

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

コンボリューション定理

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

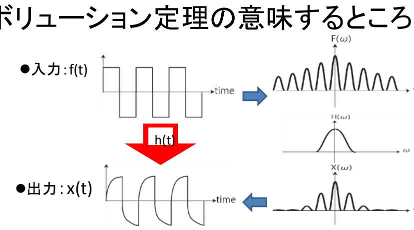
フーリエ逆変換　フーリエ変換

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

簡略化のため次のようにも表記

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

コンボリューション定理の意味するところ(1)

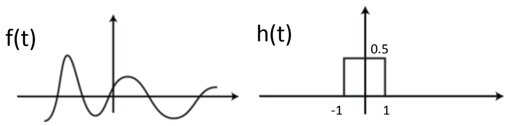


$x(t)=\int_{-\infty}^{\infty} f(\tau)h(t-\tau)d\tau$　　$X(\omega)=F(\omega)H(\omega)$

- 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)

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

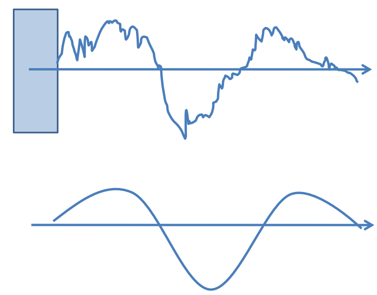


例えば，h(t)=0.5 (-1<t<1)なら，

$$x(t)=$$

これは，f(t)を平均化していくフィルタ

平均化？

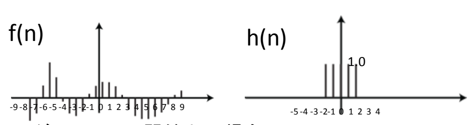


ノイズを「ならし」て大域的な特徴をつかむ

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)+…



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

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

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

$$=$$

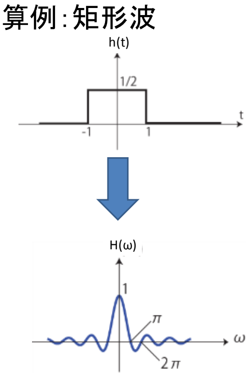
$$=\left[\frac{1}{-j2\omega}\exp(-j\omega t)\right]_{-1}^{1}$$

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

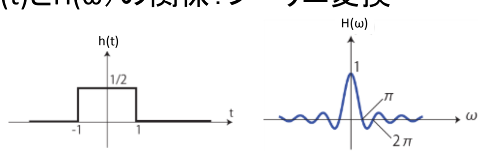
$$=\frac{1}{-j2\omega}(\cos(\omega)-j\sin(\omega)-\cos(\omega)-j\sin(\omega))$$

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

$$=$$



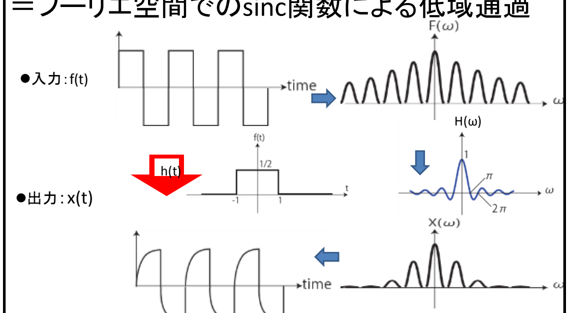
h(t)とH(ω)の関係：フーリエ変換



つまり，h(t)のフーリエ変換H(ω)は，大雑把には
「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」
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」

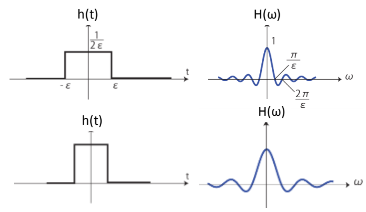
実時間での矩形波による平均化
＝フーリエ空間でのsinc関数による低域通過



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

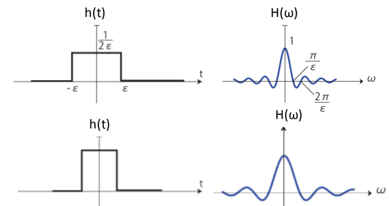
（復習）矩形波の幅が変わると？

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



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

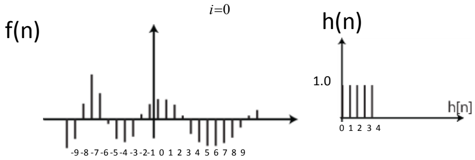


矩形波の幅を狭くする ⇒ フーリエ変換結果は幅広に

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

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

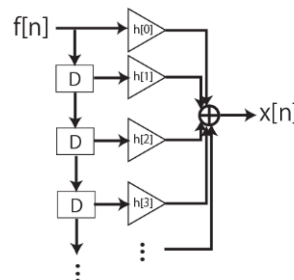


i=0から始める：未来のデータ が使えないことを意味する.
この例は，元データf(n)を，４個平均して出力する.

•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

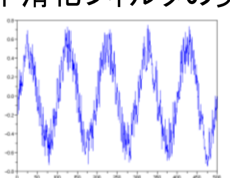
$x(n)=$



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.

平滑化フィルタの実例（１）

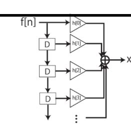


元の信号に
高周波ノイズが含まれている.

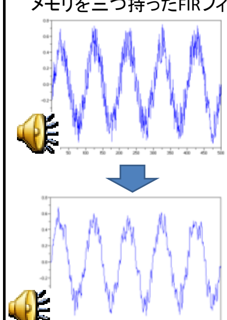
Scilabコード例

```
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

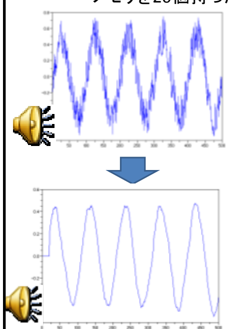


Scilabコード例

```
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)

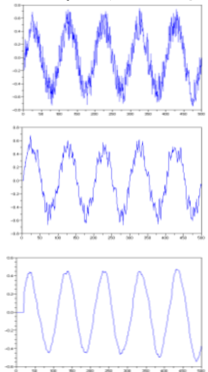
メモリを20個持ったFIRフィルタによって平滑化

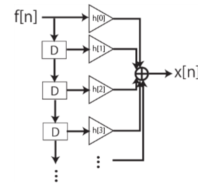


Scilabコード例

```
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));
```

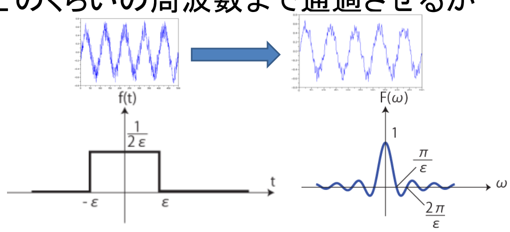
FIRフィルタによる平滑化の効果と弊害





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

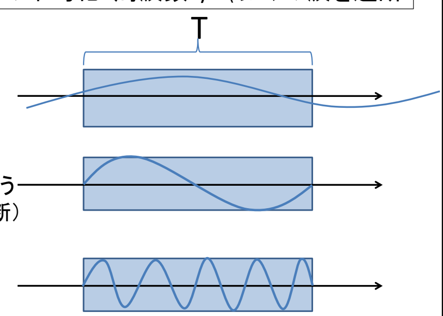


幅2εの矩形波のフーリエ変換：角周波数π/εで0

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.

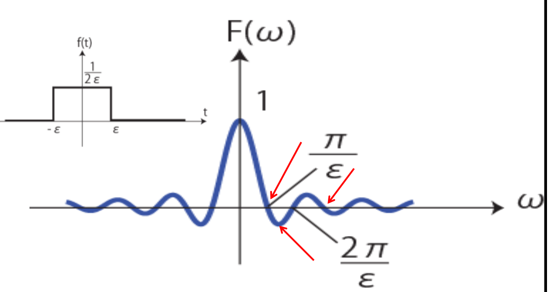


ほぼ通過

打ち消し合う
(完全に遮断)

ほぼ遮断

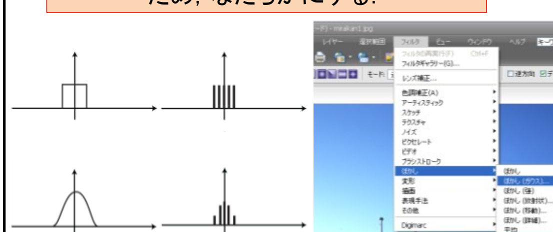
単純平均化によるローパスの落とし穴



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.

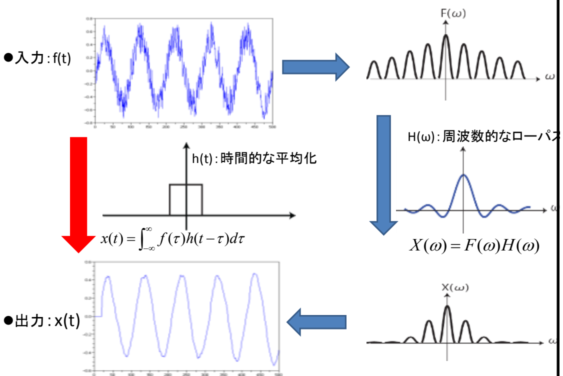
実際のローパス

周波数空間での周期的ふるまいを無くすため，なだらかにする.



画像の世界では．．．「ガウスぼかし」

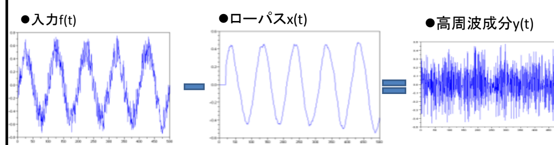
平均化によるローパスフィルタ：まとめ



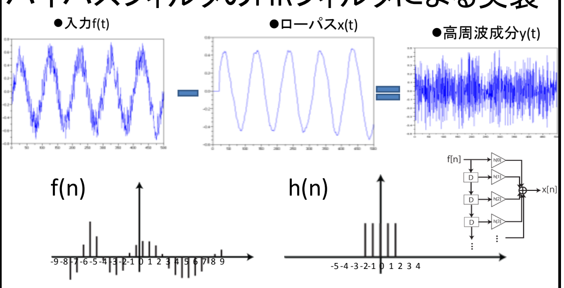
逆に高い周波数成分だけ取り出すには？

●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？



ハイパスフィルタのFIRフィルタによる実装

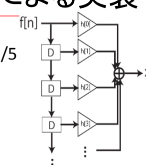


ローパス例：x(n)=(f(n+2) + f(n+1) + f(n) + f(n-1) + f(n-2))/5
ハイパス例：y(n)=f(n)-x(n)
=-1/5・f(n+2) -1/5・f(n+1) +4/5f(n) -1/5・f(n-1) -1/5・f(n-2)

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：

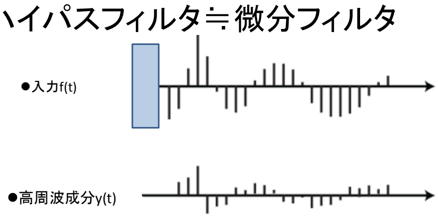
x(n)=(f(n) + f(n-1))/2

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



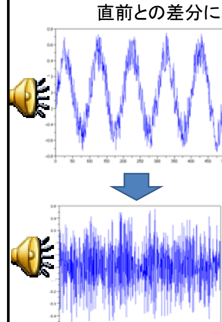
y(1) = (f(1) - f(0))/2
y(2) = (f(2) - f(1))/2
y(3) = (f(3) - f(2))/2...

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	＝	平滑化フィルタ
ハイパスフィルタ＝高域通過フィルタ	＝	微分フィルタ

ハイパスフィルタの例

直前との差分によってハイパス

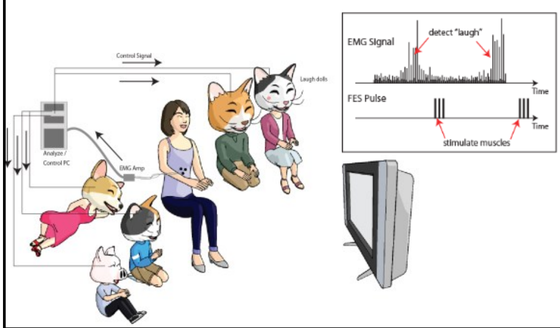


Scilabコード例

```
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));
```

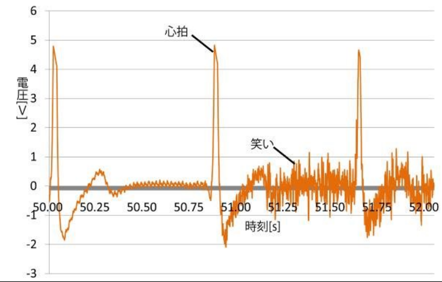
フィルタリング．．．研究の現場で

筋電計測による笑いの検出→増幅は可能か？

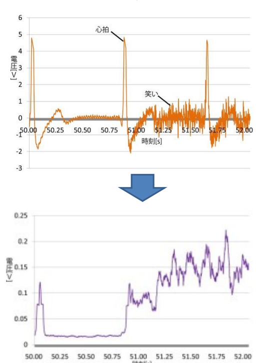


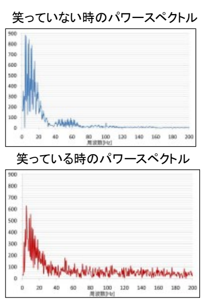
研究の現場で

筋電計測：
●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

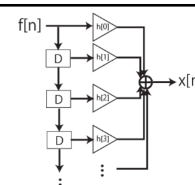




（1）ハイパスフィルタで笑い成分を抽出
（2）絶対値化フィルタで正の値に変換
（3）ローパスフィルタで笑い領域を確定

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

レポート課題1

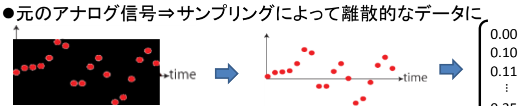
適当なwaveファイルに対して次の三つの操作を行う．
（1）FIRフィルタによるローパスフィルタをかけて音をくもらせる．
（2）FIRフィルタによる適当なハイパスフィルタをかけて音をとがらせる．
（3）エコーを掛けてカラオケのようにする．

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．）

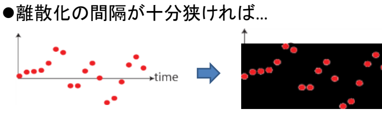
※注：Waveファイルの形式によってはScilabで扱えない場合があります。その場合は、例えばWindows標準のWaveサウンドファイル（Windows開始音等）を使うとうまくいきます。C:¥Windows¥Mediaの下にあります。

PCで信号を扱う＝離散化

●元のアナログ信号⇒サンプリングによって離散的なデータに

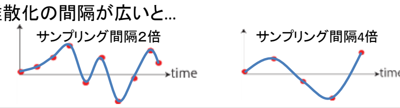


●離散化の間隔が十分狭ければ...



元に戻せそう

●離散化の間隔が広いと...



元に戻せなそう

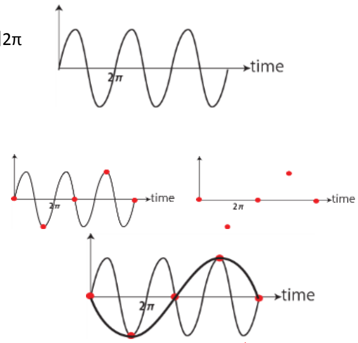
この違いはなんだろうか？　「元に戻す」とはなんだろうか？

元に戻せない（＝元が推測できない）場合

元の信号：sin(x)，周期2π

離散化の間隔　3/2 π

なめらかに結ぶと…



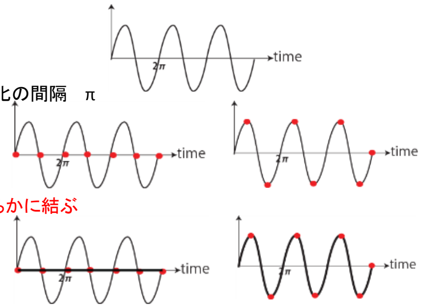
元と全く異なる波形となる＝エリアシング

離散化に際して：ナイキスト周波数

元の信号：sin(x)，周期2π

離散化の間隔　π

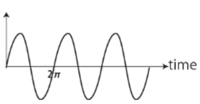
なめらかに結ぶ



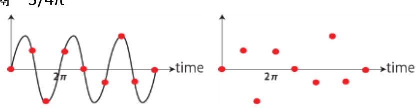
うまくいく場合と、うまくいかない場合がありそう

離散化に際して：ナイキスト周波数未満

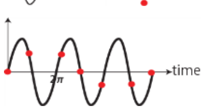
元の信号：sin(x)，周期2π



離散化の間隔　3/4π



正弦波でなめらかに結ぶ



元の波形が再現できる！！

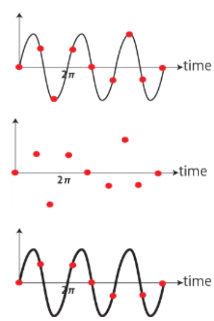
サンプリング定理（標本化定理）

元信号に含まれる<u>最高周波数</u>の，

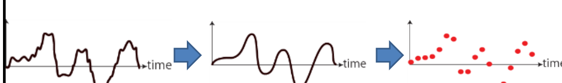
倍より高い周波数でサンプリング（標本化）していれば，

元の信号はサンプリング点から完全に再生できる．

$\boxed{\text{倍の周波数＝ナイキスト周波数}}$



サンプリング定理（標本化定理）



逆に，エリアシングを生じないために，

サンプリング周波数の半分以上の周波数は，**<u>あらかじめカット</u>**する必要がある．（後でカットしても意味無し！）

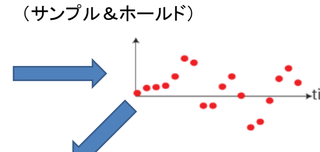
カットしないとエリアシングを生じ，偽の低い周波数が観察される．
（例）蛍光灯下の扇風機，テレビ画面のビデオ撮影

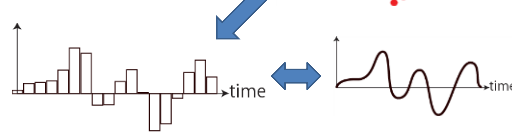
カットはアナログ回路によるローパスフィルタなどを用いる事が多い

サンプリングデータを元に戻す（再生）とは？

一番簡単な方法：サンプリングされたデータを、単純に電圧出力する（サンプル＆ホールド）

$$\begin{pmatrix}0.00\\0.10\\0.11\\\vdots\\0.35\end{pmatrix}$$

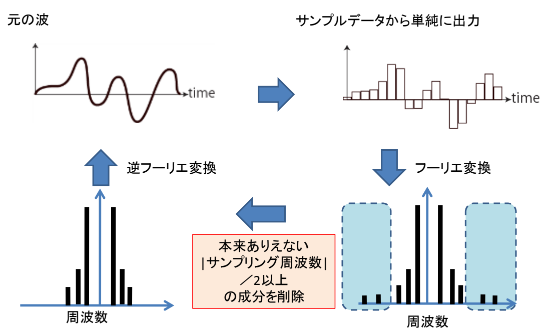




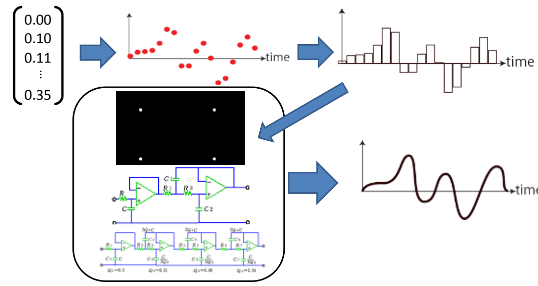
大体同じ。でも微妙な違い

元の波が、ナイキスト周波数未満の成分しかないとすると、
単純に、「<u>高い周波数をカット</u>すれば元に戻る」はず

元の波に含まれない周波数をカット（イメージ）

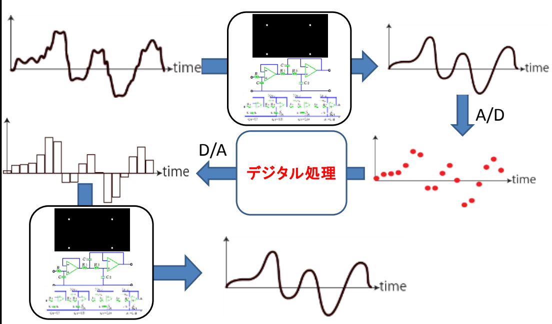


サンプリングデータ（デジタルデータ）からアナログ波形の出力の実際



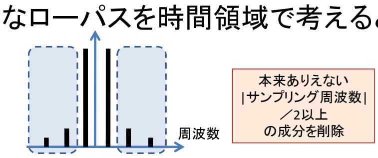
実際には**アナログ回路**で高周波成分をカットする場合がほとんど
カット周波数＝サンプリング周波数の半分

「デジタル」処理のための「アナログ」処理まとめ



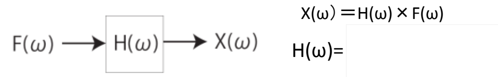
①A/D前にサンプリング定理を満たすためのローパス
②D/A後にサンプリング定理を満たすためのローパス

理想的なローパスを時間領域で考えると？



信号のフーリエ変換F(ω)にフィルタH(ω)をかけることを意味する

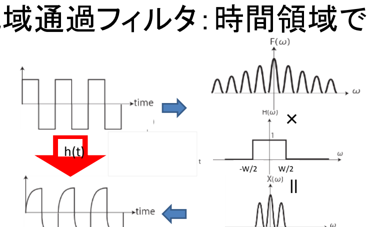
X(ω)＝H(ω)×F(ω)



理想的低域通過フィルタ：時間領域では？

●入力：f(t)

●出力：x(t)

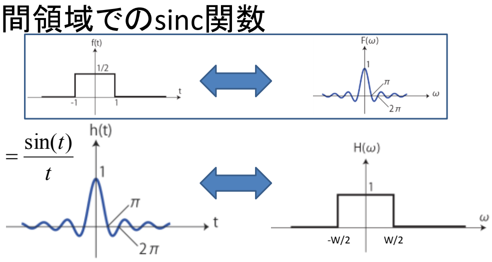


H(ω)= {1 (|ω|＜W/2), 0 (|ω|≧W/2)

逆フーリエ変換でh(t)を求めてみる

$h(t)=$

時間領域でのsinc関数

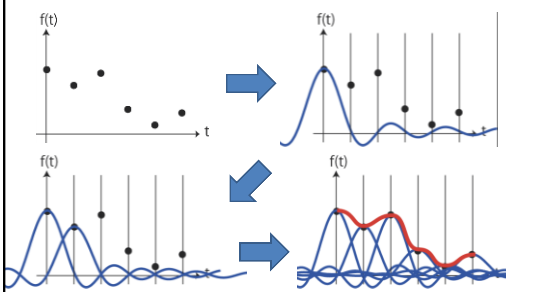


$$h(t)=\frac{\sin(t)}{t}$$

周波数領域での理想的な低域通過フィルタは、
時間領域でのsinc関数に他ならない

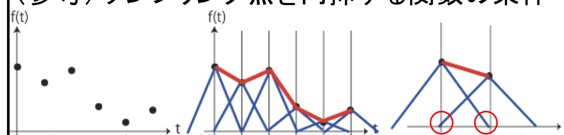
ただし無限に長いフィルタになるので、
結局矩形波（＝平均化）や、よりなだらかな波形が用いられる。

（参考）サンプリング点を「なだらかに内挿する」関数としてのsinc関数

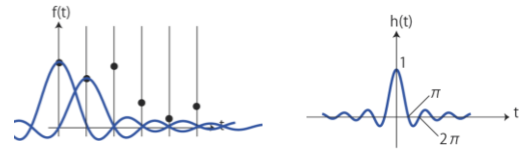


Sinc関数での「畳込み積分」:
サンプリング点ごとにsinc関数を重ねあわせていく操作に相当

（参考）サンプリング点を内挿する関数の条件

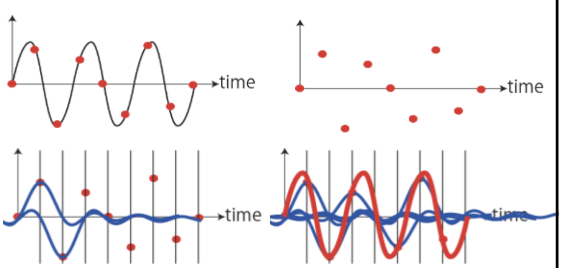


最も簡単な内挿＝線形補間。この場合基礎となる関数は孤立三角波
各関数の合算結果がサンプリング点を通過するためには、
基礎となる関数が、サンプリング点で0にならなければならない。



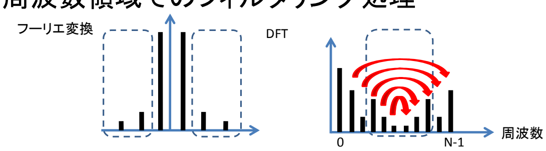
Sinc関数はこの条件を満たしている。

（参考）サンプリング定理ぎりぎりの波形も

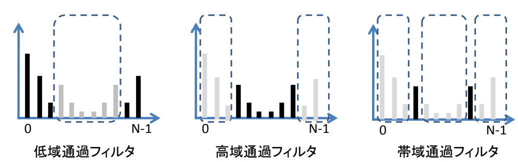


サンプリング定理ギリギリの波形のサンプリング結果も、
Sinc関数で理論通りに内挿するとちゃんと元に戻る

周波数領域でのフィルタリング処理



フーリエ変換に対するフィルタリング：原点対称
DFTに対するフィルタリング：中心対称に作用させる必要



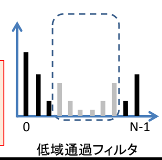
レポート課題2

低域通過フィルタによって三角波を正弦波にする。

（1：時間領域での処理）レポート課題1と同様のFIRフィルタをかけ、波形が正弦波に近づいていくことを観察せよ

（2：周波数空間での処理）三角波のフーリエ変換結果に対して、周波数領域で低域を通過させた後、逆フーリエ変換で波形を元に戻せ。

理解してほしいこと：時間領域での処理（畳込み積分）と周波数空間での処理が同じ結果を生むことを認識。



レポート課題2(2) 参考（ほぼ答え）

```
wave=[-49:50];    //一周期100の三角波
wave = [wave,wave,wave,wave,wave]; //5回繰り返す。つまり500要素の波形
plot(wave);
fourier = fft(wave);  //フーリエ変換。500要素のベクトル

//パワースペクトルを計算
//power_spec = fourier .* conj(fourier);
//plot(power_spec);    //計算結果を表示

//フーリエ変換結果から高域を取り除く。どこからどこまで取り除くかは、パワースペクトルの観察で見極める。DFT結果に対しては左右対称に取り除くことに注意
//Scilabでは配列の添字が1から始まることに注意
for i=
  fourier(i)=0;
end

wave2=ifft(fourier); //逆フーリエ変換
plot(wave2);
```