10 FIR フィルタ

10.1 線型システム

ディジタル信号処理の分野では、何らかの入力信号を出力信号に変換するものを「システム」と呼ぶ。

これまでに紹介した処理で、例えば、x_1 という波と x_2 という波を足し合わせて y_1 という波を作成する、という場合は、システムは「2 つの入力を加算する」という変換を行う。(x_1, x_2 が入力で、y_1 が出力) これは式では次のように書く。

$$y_1 = x_1 + x_2 \tag{5}$$

この入力信号、出力信号は、実際にはディジタルデータ (離散データ) である。そのことを反映する場合には、次のように書くのが普通である。

$$y_1[n] = x_1[n] + x_2[n] \tag{6}$$

n はデータの添字をあらわす。音声データの場合は、時刻と同じようなものだが、n と書くような場合は添字が整数であることを意味することが多い。

別の例として、入力信号 x_3 を a 倍して y_2 を作成するという場合は、次のように書ける。

$$y_2[n] = ax_3[n] \tag{7}$$

システムの入力が x_i であり、出力が y_i であるとする。このとき、入出力の関係が次のように x_i, y_i に関する一次式で表されるとする。

$$a_1y_1 + a_2y_2 = b_1x_1 + b_2x_2 \tag{8}$$

このように一次式で表されるシステムを線型 (線形) システムとよぶ。

10.2 遅延演算

線型システムの基本的な演算として、信号を定数倍する演算や、加算する演算を紹介してきた。もう 1 つの基本的な演算として、遅延がある。遅延は、入力を単位時間の定数倍だけ遅らせる演算である。(離散データにおいては、単位時間とは、サンプリング周期のことであり、添字を 1 変化させることに対応する。)

遅延を式でどのように表すかを考えてみる。1 単位時間遅らせて出力信号とする場合は、時刻 n の入力が 時刻 $n+1$ の出力となるということである。入力信号を x、出力信号を y で表すと、次のようになる。

$$y[n+1] = x[n] \tag{9}$$

遅らせるだけだと余り意味はないが、遅らせた信号を元の信号に足し合わせると様々な効果を得られる。

ソースコード 19: 遅延演算を用いたフィルタリング

```
1  >> fs=8000;
2  >> t=0:1/fs:1-1/fs;
3  >> s=sin(2*pi*800*t)+sin(2*pi*500*t);
4  >> soundsc(s,fs)
5  >> r=1:100;
6  >> subplot(3,1,1); plot(s(r));
7  >> sd=wshift(1,s,-5);
8  >> subplot(3,1,2); plot(sd(r));
9  >> soundsc(sd,fs)
10 >> ss=s+sd;
11 >> subplot(3,1,3); plot(ss(r));
12 >> soundsc(ss,fs)
```

6 行目などの subplot は複数のプロットを並べる関数である (詳しい使い方は help subplot で調べよ)。また、7 行目の wshift は信号を遅らせるのに使っている。help wshift をすればわかるように、ただ遅らせているわけではない。ここでは、s の長さを変化させないために wshift を利用している。

プロットを見ればわかるように、この例では、5 点遅らせて足し合わせることで、2 つの成分を含んでいた信号 s が、正弦波のようになった。つまり、片方の成分が除去された。

このように遅延演算と加算、定数倍演算を上手く組み合わせると、ある成分の除去や強調ができる。一般にはある成分を除去することが多いため、このような処理をディジタルフィルタと呼ぶ。

10.3 移動平均フィルタ

簡単だけれども、画像処理や株価の計算など非常に様々な分野で利用されるフィルタに移動平均フィルタとよばれるものがある。

入力の連続する 3 点の平均を出力とするフィルタを考えてみる。式であらわすと次のようになる。

$$y[n] = \frac{x[n] + x[n-1] + x[n-2]}{3} \tag{10}$$

このフィルタの効果を音を対象に調べてみる。

まず、雑音を作る。

ソースコード 20: 白色雑音の生成

```
>> fs=8000;
>> t=0:1/fs:1;
>> r=0.8*(-1+2*rand(size(t)));
>> n=1:100;
>> plot(t(n),r(n))
>> sound(r,fs)
```

ここでは、白色雑音 (ホワイトノイズ) とよばれる雑音を 1 秒間生成している。信号処理では、雑音とは、ランダム性のある (周期性のない) 信号である。白色雑音とは一様分布の乱数列である。

MATLAB では、音データ y は、$-1 \leq y \leq 1$ を満たさなければ歪んでしまう。そのような条件を満たすように、3 行目では、$(0, 1)$ 区間の一様乱数を生成する rand 関数を利用して、$(-0.8, 0.8)$ 区間の一様乱数を 1 秒間分生成している (どういうベクトルになっているかは、rand 関数と size 関数を help コマンドで調べて理解すること)。

この雑音 r を少しだけ正弦波に加えると雑音混じりの音が生成できる。

ソースコード 21: 雑音混じりの正弦波の生成

```
>> s=sin(2*pi*440*t);
>> sn=0.8*s+0.1*r;
>> n=1:100;
>> plot(t(n),sn(n),t(n),0.8*s(n))
>> sound(sn,fs)
```

4 行目の plot 関数では、sn と s を重ねてプロットしている。

この雑音混じりの正弦波を 3 点移動平均フィルタにかけてみる。そうするためには、雑音混じりの信号 sn を入力にして出力を求めればよい。まずは、for ループを使って求めてみる。

ソースコード 22: for ループを用いた 3 点移動平均フィルタ

```
>> pn=3;
>> for i=pn:length(t),
y(i)=mean(sn(i+((1:pn)-pn)));
end
>> plot(t(n),sn(n),t(n),y(n),t(n),0.8*s(n))
>> sound(y,fs)
>> sound(s,fs)
>> sound(sn,fs)
```

式 (10) を書き直すと次のようになる。

$$y[n] = \frac{1}{3}x[n] + \frac{1}{3}x[n-1] + \frac{1}{3}x[n-2] \tag{11}$$

この式は、遅延演算と定数倍と加算だけで構成されていることがわかる。遅延演算は入力を遅らせるだけなので、いくつ遅らせるかがわかれば十分である。そこで、上記の演算を [1/3　1/3　1/3] と遅延の位置に定数倍の係数を与えて作成できるベクトルで表わすことがある。この係数ベクトルを利用して簡単にフィルタをかける演算が用意されている。その演算が畳み込み (コンボリューション) である。MATLAB では conv 関数で計算できる。

ソースコード 23: 畳み込みを用いたフィルタリング

```
1 >> h=ones(1,3)/3;
2 >> y2=conv(h, sn);
3 >> plot(t(n),y2(n),t(n),y(n))
```

畳み込み演算を式で書くときには、$*$ という記号を使って、$h * s_n$ のように書く。

11　インパルス応答

11.1　インパルス応答

システムの効果を示すために用いられる方法にインパルス応答がある。応答とは、システムの入力に対して得られる反応 (出力) のことである。インパルス応答とは、入力をインパルスとしたときの出力である。

次の式は単位インパルスを表す。

$$x[n] = \begin{cases} 1 & (n = 0) \\ 0 & (n > 0) \end{cases} \tag{12}$$

このように 1 点だけ値がある信号をインパルスという。

1024 点の長さを持つインパルスは MATLAB では次のように作る。

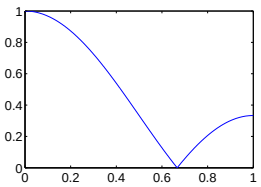
```
>> imp([1 1024]) = [1 0];
```

3 点の移動平均フィルタのインパルス応答は次のように求められる。

```
>> h=ones(1,3)/3;
>> h_imp=conv(h,imp)
>> stem(h_imp)
```

インパルス応答のスペクトルのことをフィルタの周波数応答とよぶ (周波数特性ともいう)。

3 点の移動平均フィルタの周波数応答は次の図のようになる。(横軸は正規化周波数)

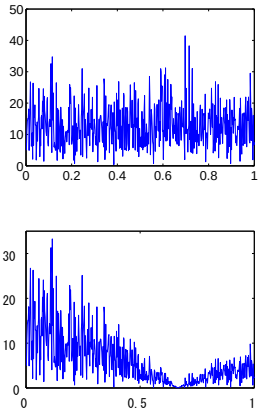


時間領域の畳み込みは、周波数領域では乗算となる。つまり、x のスペクトルを X、s_n のスペクトルを S_n とすると、$x * s_n$ のスペクトルは $X \cdot S_n$ となる。

ソースコード 24: 周波数領域でのフィルタリング

```
1 >> S_imp=fft(h_imp,1024);       % フィルタのスペクトル (上図)
2 >> wn=0.8*(-1+2*rand(1,1024)); % 白色雑音
3 >> WN=fft(wn);                  % 白色雑音のスペクトル
4 >> WN_filtered=abs(WN.*S_imp); % 白色雑音にフィルタをかける
```

3 行目、4 行目のスペクトルをプロットすると以下のようになる。(ただし、雑音は生成するたびに値が変化するので、全く同じにはならない。)



フィルタのスペクトルの形状にあわせて雑音のスペクトルの概形が変化しているのがわかる。この処理は、4 行目の `.*` で実現している。

12 IIR フィルタ

システムでは、出力をもう一度入力として利用することで効率よい処理を行えることがある。

$$y[n] = -0.5y[n-5] + x[n] \tag{13}$$

という式を考える。この式は

$$y[n] = -0.5x[n-5] + x[n] \tag{14}$$

と形は似ているが、得られる出力は全然違う。このことはインパルス応答を調べることで明らかになる。

例えば、次の式で表わされる 10 点の単位インパルスに対し、式 (13) と 式 (14) のインパルス応答を計算してみるとよい。

$$x[n] = \begin{cases} 1 & (n = 0) \\ 0 & (1 \leq n \leq 10) \end{cases} \tag{15}$$

インパルス応答は、漸化式にしたがって出力の値を計算すれば求まる。例えば、式 (15) を式 (13) にあてはめると次のようになる。

$$y[0] = -0.5y[-5] + x[0] = -0.5 \cdot 0 + 1 = 1$$
$$y[1] = -0.5y[-4] + x[1] = -0.5 \cdot 0 + 0 = 0$$
$$\cdots$$

式 (13) のように出力を入力に用いることをフィードバックと呼ぶ。フィードバックがあるとインパルス応答は無限に続くため、このようなタイプのフィルタは IIR (infinite impulse response) フィルタと呼ばれる。一方で、フィードバックがないフィルタのインパルス応答は有限なので、FIR (finite impulse response) フィルタと呼ばれる。

IIR フィルタの係数は フィードバックを考慮しないといけないため、FIR フィルタの係数とは同様には表現できない。そこで、入力に関する項は右辺、出力に関する項は左辺とまとめる。例えば、式 (13) は次のように変形する。

$$y[n] + 0.5y[n-5] = x[n] \tag{16}$$

この両辺の係数を別々に、左辺を出力の係数 (ベクトル `b`)、右辺を入力の係数 (`a`) で表すとする。この 2 つのベクトルを用いると、`filter` 関数でフィルタを入力に適用できる。

```
>> ir_iir = filter(b,a,imp);
```

FIR フィルタの場合にも、filter 関数を使える。FIR フィルタの場合は、漸化式の左辺は、必ず $y[n] = \ldots$ となるので a = 1 となる。

13 フィルタ設計のツール

フィルタ係数をうまく設定することで様々な効果を得ることができる。フィルタは、その効果により分類されている。最もよく用いられるフィルタとして、ある周波数の範囲 (帯域とよぶ) を通さない効果を持つものがある。MATLAB では、典型的なフィルタについては、求める効果から係数列を計算する関数が用意されている。

例えば、ある周波数より高い周波数を通さないフィルタは LPF(ローパスフィルタ) と呼ばれる。サンプリング周波数が 8kHz の信号に対し、2kHz 以上の成分を通さない LPF のフィルタ係数は fir1 という関数を用いて次のように計算できる。

ソースコード 25: fir1 を用いた LPF の設計

```
>> h_lp2k=fir1(40,0.5);
>> stem(h_lp2k)
```

かなり複雑な係数であることがわかる。このフィルタを**ソースコード 21** の sn にかけるには、次のように filter 関数を使えばよい。

```
>> snf=filter(h_lp2k,1,sn);
```

LPF は IIR フィルタでも設計できる。

ソースコード 26: butter を用いた LPF の設計

```
>> [b,a]=butter(10,0.5);
```