2　ディジタルフィルタ

これまで、ディジタル信号の振幅に係数を掛ける処理と、信号と信号を加える処理をみてきた。もう一つの重要な処理が、信号を遅らせることである。遅らせることによる効果を見てみる。

```
>> t = 0:1/8000:1;
>> s = sin(2*pi*800*t) + sin(2*pi*500*t);
>> plot(s(1:100))
>> soundsc(s,8000)
```

s を 5 点 (5/8000 秒) 遅らせた信号として sd を作る。

```
>> sd = [0 0 0 0 0 s]; % 先頭に 5 つ 0 を挿入
>> n = 1:100;
>> plot(n,s(n),n,sd(n));
```

s と sd を足し合わせる。

```
>> ss = s + sd(1:length(s));
>> plot(n,ss(n));
```

最初の部分を除いて、一つの成分が除去されていることがわかる。

練習 7 何の成分が除去されているか?

練習 8 別の成分を除去するためには、何点遅らせたものを加えればよいか?

このように、遅れを上手く利用すると音を変化させることができる。

ディジタル信号を遅らせたり、加えたり、ある係数を掛けたりすることで音を変化させるディジタルフィルタを構成することができる。

上記と同じ処理をおこなうディジタルフィルタは h = [1 0 0 0 0 1] という数列であらわされる。このフィルタを信号 s に適用する。

```
>> h = [1 0 0 0 0 1];
>> ss2 = conv(h,s);
>> plot(n, ss(n), n, ss2(n));
```

conv は畳み込みという計算をおこなう。

練習 9 conv([1 3], [2 4]) などの計算を試してみて、conv とはどのような計算かを確認せよ。help conv などの情報も利用せよ。

h という系列をどう設計するかで、様々な効果が得られる。

```
>> n = 1:100;
>> t = 0:1/8000:1;
>> nz = randn(1, length(t));
>> plot(n, nz(n));
>> soundsc(nz, 8000);
>> s = sin(2*pi*440*t);
>> sn = s + 0.1 * nz;
>> plot(n, sn(n));
>> soundsc(sn, 8000);
```

sn は雑音まじりの音である。

```
>> h = ones(1,3)/3
>> soundsc(conv(sn,h),8000);
```

雑音が軽減されたことがわかる。

練習 10 h の長さを長くすることで、結果として得られる音声がどのように変化するか試してみよ。

練習 11 h = [1 -1] として、上記の sn に適用してみよ。どのような音になるか?

練習 12 これらの変化を spectrogram で観察し、どのような変化がおきているか確認せよ。

最もよく用いられるフィルタの一つが、ある周波数の範囲 (帯域とよぶ) を強めたり弱めたりする処理である。前回のダウンサンプリングでは、高い周波数が残っていては困る。そこで、ある周波数より、低い周波数だけを残して、高い周波数を除去する LPF(ローパスフィルタ、低域だけを通過 (パス) させる) を用いる。

サンプリング周波数 8kHz で、2kHz 以下の低域だけを通過させる LPF の数列は、以下のように計算できる。

```
>> h = fir1(40, 0.5); % 第一引数はフィルタの点数
                      % 第二引数は周波数の指定
                      % ナイキスト周波数が 1 となる
>> stem(h);
```

数列はかなり複雑であることがわかる。

練習 13 h を上記の sn に適用してみよ。どのような音になるか?

練習 14 fir1 を用いて、550Hz 以下の低域だけを通過させる LPF を設計せよ。その LPF を sn に適用せよ。

練習 15 fir1 を用いて、ある周波数以上の高域を通過させるフィルタも設計できる。高域を通過させるフィルタは HPF(ハイパスフィルタ) という。450Hz 以上を通過させる HPF を設計せよ。sn に練習 14 の LPF と HPF を両方適用してみよ。

練習 16 いろいろな HPF、LPF を設計し、いろいろな音に適用してみよ。