第4章

FIRフィルタ

DSPの応用の中で，もっとも基本的でしかも重要なのは，ディジタル・フィルタです．ディジタル・フィルタを大別すると，FIRフィルタとIIRフィルタに分類できます．本章ではFIRフィルタを扱い，IIRフィルタは第5章で扱います．

本章では，まず4.1節でFIRフィルタと，その代表的な構成方法について説明します．次に4.2節で，フィルタの設計方法について簡単にふれた後，FIRフィルタのプログラムを作成します．4.3節ではC6713 DSK用に浮動小数点演算を使用したプログラムを作成し，4.4節ではC6416 DSK用に固定小数点演算を使用したプログラムを作成します．この二つの節で作成するプログラムは，C++言語のクラスを使わないで作成します．最後の4.5節では，C++言語のクラスを使ったプログラムを作成します．

4.1 FIRフィルタとその構成法

(1) FIRフィルタとは

FIRとはFinite Impulse Responseの略で，インパルス応答の継続時間が有限であるフィルタがFIRフィルタです．インパルス応答とは，入力信号として式(4.1)で示される単位インパルス信号$\delta[n]$を入力したときの出力信号のことです．

$$\delta[n]=\begin{cases}1, & n=0\\ 0, & n\neq 0\end{cases} \quad \cdots\cdots(4.1)$$

FIRフィルタの入出力信号の関係を表す差分方程式は，次の式で表されます．

$$y[n]=\sum_{k=0}^{M} h_k x[n-k] \quad \cdots\cdots(4.2)$$

この式でh_k $(k=0,1,\cdots,M)$はフィルタの特性を決める係数です．また，Mはこのフィルタの次数[注4.1]と呼ばれています．

式(4.2)で，インパルス応答を考えてみるため，$x[n]=\delta[n]$とおいてみます．そうすると，$n>M$に対して，その出力は必ず0になることが，式(4.2)から明らかです．したがって，インパルス応答の継続時間は有限であるということがわかります．

式(4.2)の差分方程式で表されるFIRフィルタの伝達関数$H(z)$は，式(4.3)のようになります．

$$H(z)=\sum_{k=0}^{M}h_k z^{-k} \quad \cdots\cdots (4.3)$$

(2) FIRフィルタの構成法

FIRフィルタを構成するには，いくつかの方法があります．本章では，最も基本的な直接形とその転置形のプログラムを作成します．その他，縦続形，格子形などの構成法がありますが，これらはあまり使われていないので，本書では省略します．

- **直接形のFIRフィルタ**

直接形とは，式(4.2)の差分方程式をそのまま使うものです．そのブロック図を**図4.1**に示します．

- **直接形に対する転置形のFIRフィルタ**

転置形とは，元の構成に対して次の三つの操作を行ったものです．

1. 入力と出力を交換する．
2. 信号の流れる方向をすべて逆転する．
3. 加算器と分岐点を交換する．

図4.1のブロック図で表される直接形に対して，この三つの操作を行った転置形FIRフィルタのブロック図を**図4.2**に示します．このブロック図に対応する差分方程式は，次のようになります．

$$\begin{cases} u_M[n]=h_M x[n] \\ u_{M-1}[n]=h_{M-1}x[n]+u_M[n-1] \\ \quad\vdots \\ u_1[n]=h_1 x[n]+u_2[n-1] \\ u_0[n]=h_0 x[n]+u_1[n-1] \\ y[n]=u_0[n] \end{cases} \quad \cdots\cdots (4.4)$$

以降では，“直接形に対する転置形”とはいわずに，単に“転置形”ということにします．

4.2 FIRフィルタの設計

フィルタのプログラムを書くためには，フィルタの係数を求める必要があります．これをフィルタの設計といいます．フィルタの設計をすべて自分でやろうとすると，そのための勉強が必要になりま

注4.1：FIRフィルタの次数と係数の個数は間違えやすいので注意が必要である．通常，FIRフィルタの次数とは，フィルタの計算に使う入力信号の中で，最も過去の信号が何サンプル前のものかというときの「サンプル数」に一致する．式(4.2)の場合，計算に使う入力信号で，最も過去の信号は$x[n-M]$であり，これはMサンプルだけ過去の信号なので，この場合はM次のフィルタと呼ばれる．一方，係数の数は$M+1$個になる．

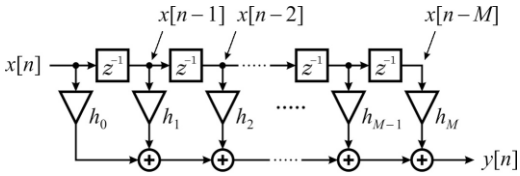

図4.1　直接形構成のFIRフィルタ

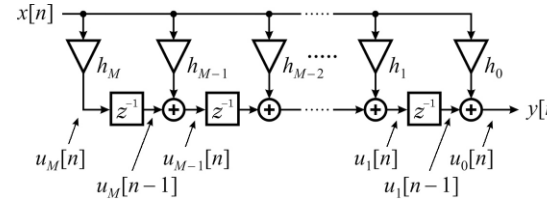

図4.2　直接形の転置形構成のFIRフィルタ

表4.1　4.3節と4.4節で作成するFIRフィルタの設計時に与えたパラメータ

次数	100	
標本化周波数〔kHz〕	48	
	帯域1（通過域）	帯域2（阻止域）
下側帯域端周波数〔kHz〕	0.0	1.6
上側帯域端周波数〔kHz〕	0.8	24
利得	1	0
重み	1	5

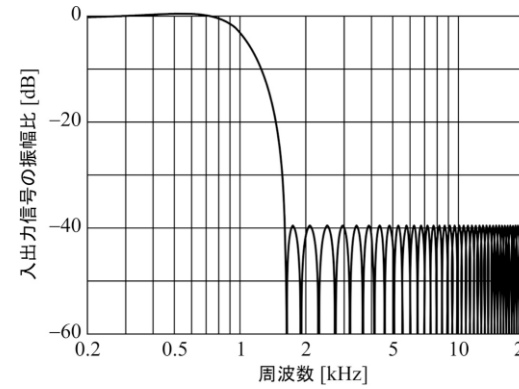

図4.3　4.3節と4.4節で作成するFIRフィルタの振幅特性

すが，それなりの数学的な知識が要求されるため，それほど簡単ではありません．また，フィルタの設計方法を理解できたとしても，設計のためのプログラムを作成するのは，かなりやっかいです．

そこで，筆者が作成したフィルタ設計のためのツールFIR_Remezを，本書付属CD-ROMに収録しました．FIRフィルタを設計する場合は，このFIR_Remezを使います．使い方は付録Eで説明しているので，そちらを参考にしてください．

本章の4.3節と4.4節で作成するFIRフィルタの係数は，このFIR_Remezを使い，**表4.1**のパラメータを与えて求めたものです．通過域は0～0.8 kHzと狭くなっています．この通過域の値は，作成したフィルタに音楽などを通して，その出力を耳で聞いた場合に，フィルタの効果を確認できるようにという観点で選んだものです．

このFIRフィルタの振幅特性を**図4.3**に示します．第3章で示した振幅特性の図の周波数の目盛りはリニア・スケールでしたが，この図はログ・スケールになっているので注意してください．

4.3　FIRフィルタのプログラム（浮動小数点演算を使用）

ここでは，C6713 DSK用に，浮動小数点演算を使ったプログラムを作成します．作成するのは，直接形とその転置形のプログラムです．

(1) 直接形のプログラム

● プログラムの作成

プログラム(FIR100_Direct_67x.cpp)を**リスト4.1**に示します．配列hmに格納されているのがフィルタの係数ですが，紙面の節約のため，一部分を省略しています．省略した部分については，本書付属のCD-ROMを参照してください．

main()関数では，まず過去の入力信号を格納しておくためのバッファとして使っている配列xn

リスト4.1　直接形FIRフィルタ(C6713 DSK用)(FIR100_Direct_67x.cpp)

```
//---------------------------------------------------------------------
//      直接形FIRフィルタ（C6713 DSK用）
//          100次，低域通過フィルタ
//      作成者，著作権者： 三上直樹 2008/03/19
//---------------------------------------------------------------------
#include "myDSK6713.hpp"

const int ORDER = 100;                  // 次数
const float hm[ORDER+1] = {
      0.00559363,  0.00066187,  0.00061767,  0.00051083,  0.00033672,
      0.00009210, -0.00022442, -0.00061234, -0.00106909, -0.00158946,
     -0.00216631, -0.00278932, -0.00344579, -0.00412026, -0.00479513,

               ...
               (一部略)
               ...

     -0.00545096, -0.00479513, -0.00412026, -0.00344579, -0.00278932,
     -0.00216631, -0.00158946, -0.00106909, -0.00061234, -0.00022442,
      0.00009210,  0.00033672,  0.00051083,  0.00061767,  0.00066187,
      0.00559363};

int main()
{
    float xn[ORDER+1];                  // 入力信号用バッファ
    AIC23_6713 codec;                   // デフォルト，標本化周波数＝48kHz

    float xL, yL, xR, yR;

// 入力信号用バッファをクリア
    for (int k=0; k<=ORDER; k++) xn[k] = 0.0;

    while(true)
    {
        codec.Read(xL, xR);             // 入力
//---------------------------------------------------------------------
// 左チャネルの処理
        xn[0] = xL;                     // 現在の入力をバッファの先頭へ格納
        yL = 0.0;
        for (int k=0; k<=ORDER; k++)    // 積和の計算
            yL = yL + hm[k]*xn[k];
        for (int k=ORDER; k>0; k--)     // 入力信号の移動
            xn[k] = xn[k-1];
// 右チャネルは何も処理を行わずに出力する
        yR = xR;
//---------------------------------------------------------------------
        codec.Write(yL, yR);            // 出力
    }
}
```

の内容を0にクリアします．次に，while文による無限ループの中で，FIRフィルタの処理が行われます．

FIRフィルタの処理では，最初に，入力信号を配列xnの先頭に格納します．次に，配列hmに格納されているフィルタの係数と配列xnに格納されている現在および過去の入力信号との間で積和の計算を行います．最後に，配列xnの入力信号の移動を行います．この信号の移動は，移動平均の場合と同じやり方になります[注4.2]．

● **プログラムの実行結果**

このプログラムを実行し，その振幅特性をWaveGeneとWaveSpectraで測定した例を**図4.4**に示します．WaveSpectraで振幅スペクトルを表示する際の横軸（周波数軸）の目盛りの設定では“Log”を選択しています．これで**図4.3**と同様に，**図4.4**でも周波数軸の目盛りがログ・スケールで表示されています．

(2) 転置形のプログラム

転置形FIRフィルタのプログラムを**リスト4.2**（FIR100_Transposed_67x.cpp）に示します．係数は**リスト4.1**とまったく同じなので，省略します．

転置形の場合は，直接形と違い，入力信号の移動は必要ありません．そのため，フィルタ処理を実行する部分で，for文によるループは積和の部分の1個所のみになります．

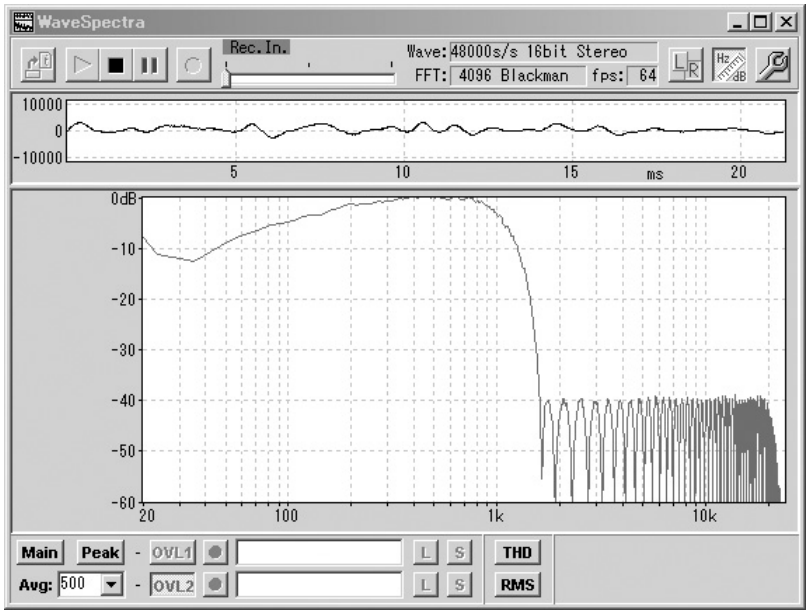

図4.4　作成した100次のFIRフィルタの振幅特性（WaveSpectraの画面）

注4.2：**図3.10**を参照のこと．