［体験セミナー］ ディジタル信号処理コース

© 2009 The MathWorks, Inc.

はじめに

本セミナーでは、はじめてMATLABをご使用になる方を対象に、デモンストレーションを通じてMATLAB/ Simulinkによるディジタル信号処理システムをご紹介すると共に、ノートPCを使って実際にMATLAB/ Simulinkの操作環境を体験していただきます。ここで取り上げるのはディジタル信号処理システムのほんの一部ですが、本セミナーがMATLAB/Simulinkによる信号処理を理解する上での参考になれば幸いです。

なお、テキスト内で実行していただくコマンドは実線の四角で囲み、次のように記述しています。

```
>> X = randn(1,1024);
>> Y = fft(X);
```

あらかじめ用意されているプログラムは、以下のように表現してあります。なお、右上の名前はファイル名です。

例題1　正規分布する乱数のパワースペクトル / samp1.m

```
X=randn(1,1024);
Y=fft(X);
A=(abs(Y)/length(Y)).^2;
plot(10*log10(A))
```

本セミナーテキストはWindows環境にインストールされたMATLAB7.2(R2006a)をベースに記述されております。UNIX環境にインストールされたMATLABとは若干インタフェースの表示が異なりますのでご了承願います。

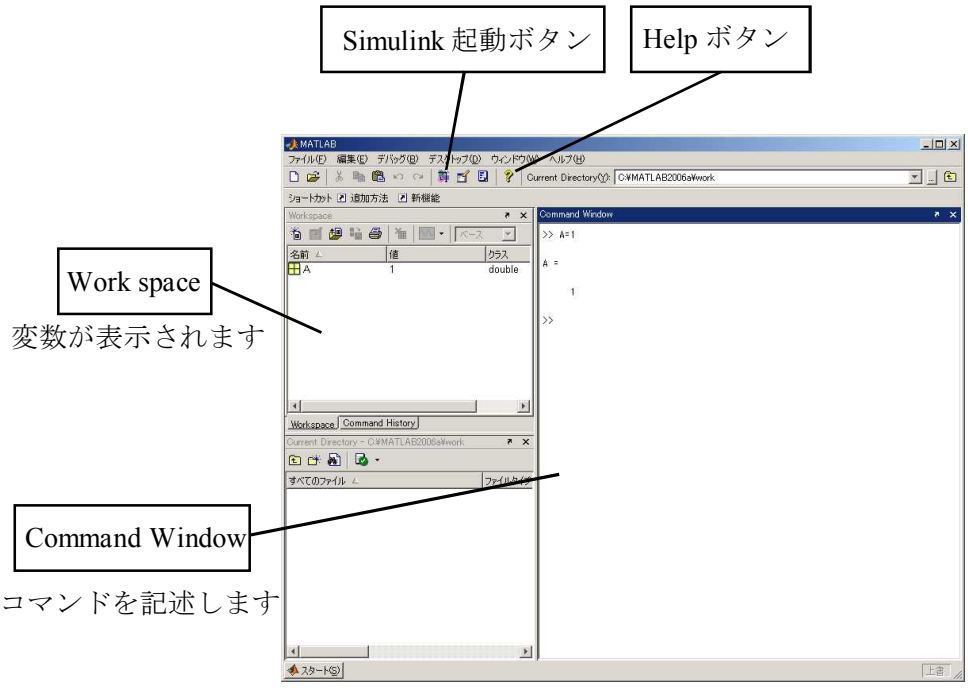


図　MATLAB操作環境

目次

MATLABによるディジタル信号処理システムの開発環境

MATLABは、あらゆる数値解析で必要とされる高機能・高精度の数値計算ファイル関数と、アルゴリズム指向の柔軟なプログラミング環境、ビジュアライゼーション機能、GUI環境の４つの基本機能を持ち、強力な数値演算機能をベースにした解析環境を提供します。

プログラミングを主とするMATLABに対して、Simulinkは関数ブロックを用いてモデルを構築しシミュレーションを行います。Simulinkは連続系、離散系、連続-離散混在系、マルチレート等汎用的なシステムをモデル化することができます。

ディジタル信号処理システム支援ツール

MATLAB製品ファミリには、ディジタル信号処理システムの開発を支援する様々なツールが用意されています。ここではその代表的なものを簡単に紹介します。

○ MATLAB
アルゴリズム開発、データ解析、ビジュアライゼーションのための高性能な計算環境

○ Simulink
システムレベル設計およびモデリングのためのグラフィカルなシミュレーション環境

○ Signal Processing Toolbox
汎用的なアナログ・ディジタル信号処理用ツール

○ Signal Processing Blockset
ディジタル信号処理システムをSimulink環境でモデル化するためのブロックライブラリ

● Filter Design Toolbox
ディジタルフィルタ設計の専用関数ツール

● Communications Toolbox / Blockset
アナログ・ディジタル通信における解析・設計・シミュレーションのためのツール

● Wavelet Toolbox
ウェーブレット解析ツール

● Image Processing Toolbox
画像処理ツール

● Simulink Fixed-Point
固定小数点演算のためのSimulinkオプション

● Stateflow
ディジタルシステムのシーケンス

● Real-Time Workshop
SimulinkモデルのC言語コード生成

○：本セミナーで使用するツール
●：信号処理で主に用いられるその他のツール

上記の他にもMATLABプロダクトファミリには多数のツールが用意されていますので、目的に合わせた選択が可能です。

第1章 スペクトル解析

Signal Processing Toolbox には、信号処理全般にわたる機能が用意されています。ここでは、Signal Processing Toolbox を用いたスペクトル解析について、その代表的な方法を取り上げるとともに、MATLAB/Simulinkの特徴である簡潔な操作環境と多彩な機能をご紹介します。

1.1 パワースペクトルの計算

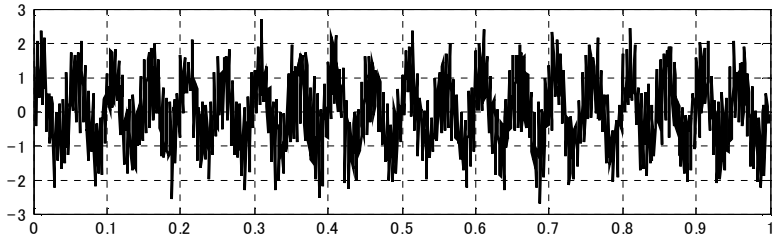
MATLAB には組み込み関数として、高速フーリエ変換（FFT)の関数が用意されています。ここでは生成した信号に対して高速フーリエ変換を行い、結果を表示します。

まず、M-file：ms1.m を起動し、次にこの M-file を Command Window 上で実行します。

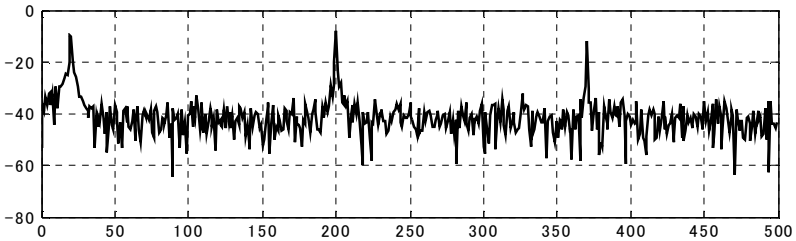
```
>> edit ms1
>> ms1
```

演習 1.1　簡単な高速フーリエ変換プログラム / ms1.m

```
t=0:0.001:1023*0.001;  ☆ポイント1                    % 時間ベクトルの生成
sig1=sin(2*pi*20*t+pi/8);                            % 元信号の生成
sig2=0.8*sin(2*pi*200*t+pi/4);
sig3=0.5*sin(2*pi*370*t+pi/2);
sig4=0.3*randn(size(t));
sig=sig1+sig2+sig3+sig4;
L=length(sig);                                       % データ長さ
fr=(1/0.001)*(0:L/2-1)/L;                            % 周波数ベクトル
x=fft(sig);  ☆ポイント2                              % FFTの計算
psd=(0.001/L)*abs(x).^2;                             % パワースペクトルの計算
                                  ☆ポイント3
subplot(2,1,1),plot(t(1:1000),sig(1:1000)),grid      % 元信号の表示
subplot(2,1,2),plot(fr,10*log10(psd(1:L/2))),grid    % 計算結果の表示
```



a)　元信号



b)　パワースペクトル

図 1.1　元信号とパワースペクトル

＜*解説*＞

☆ポイント1

- 等間隔ベクトルの生成 -　　初期値：増分：最終値

☆ポイント2

- 関数の充実 -

高速フーリエ変換の他にも信号処理に適した関数が多数用意されています。これらの関数を有効に用いることにより、効果的なプログラミング及びシミュレーションを行うことが可能です。

☆ポイント3

- グラフィックス関数の充実 -

高度なグラフィックス機能を提供します。各種グラフィックス関数を用いることにより、解析結果を簡単に表示し、検証を行うことができます。

＜参考＞

Signal Processing Toolbox には各種窓関数が用意されています。(表1.1)。これらの関数と MATLAB に用意されているフーリエ変換の関数を用いて「ピリオドグラム」、「修正ピリオドグラム」と呼ばれるパワースペクトルの推定量を求めることができます。

表 1.1　各種窓関数

bartlett	Bartlett ウィンドウ
barthannwin	修正Bartlett-Hanningウィンドウ
blackman	Blackman ウィンドウ
blackmanharris	最小4-項Blackman-Harrisウィンドウ
chebwin	Chebyshev ウィンドウ
gausswin	Gaussianウィンドウ
hamming	Hamming ウィンドウ
hann	Hann (Hanning) ウィンドウ
kaiser	Kaiser ウィンドウ
nuttallwin	Nuttall定義の最小4-項Blackman-Harrisウィンドウ
parzenwin	Parzen (de la Valle-Poussin) ウィンドウ
rectwin	矩形ウインドウ
triang	三角形のウインドウ
tukeywin	Tukeyウィンドウ

1.2 音声信号のスペクトル解析

Signal Processing Toolbox には、スペクトルの推定手法が各種用意されています（表 1.2）。取り扱うデータの性質、得たい情報の種類などに応じて適した手法を選ぶことができます。

表 1.2　代表的なスペクトル解析手法

pwelch	Welch法
pmtm	マルチテーパー法
pcov	共分散法
pmcov	修正共分散法
pyulear	Yule-Walker AR法
pbueg	Burg法
peig	固有ベクトル法
pmusic	MUSIC法

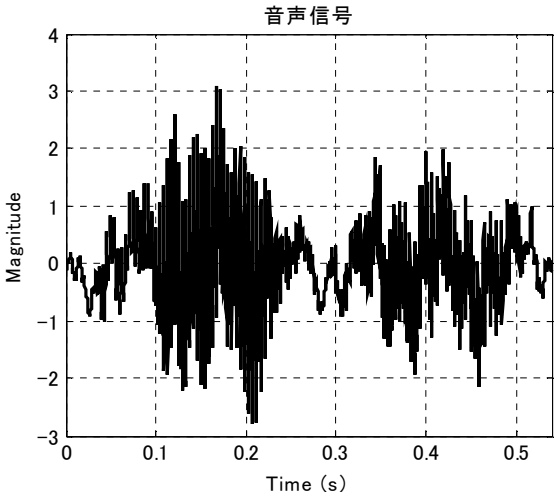
ここでは、Welch法と Yule-Walker AR法を使って音声信号のスペクトル解析を行います。また、スペクトログラムを計算して、結果を 2 次元データ、3 次元データとして表示します。

プログラム ex1_2.m を起動します。Command Window 上に以下のように入力してください。

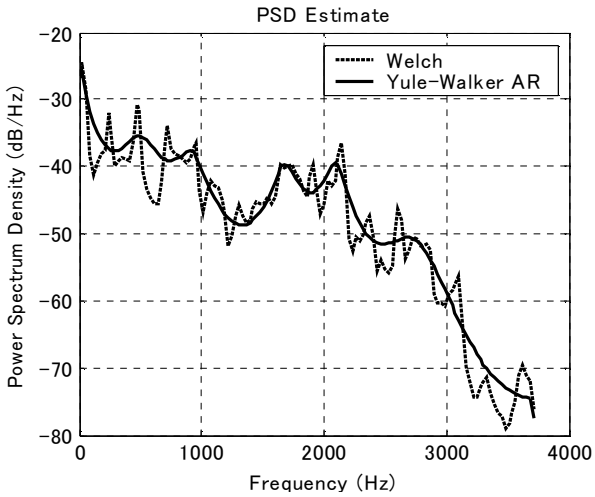
```
>> edit ex1_2
```

Command Window 上に以下のように入力して、プログラムを実行します。

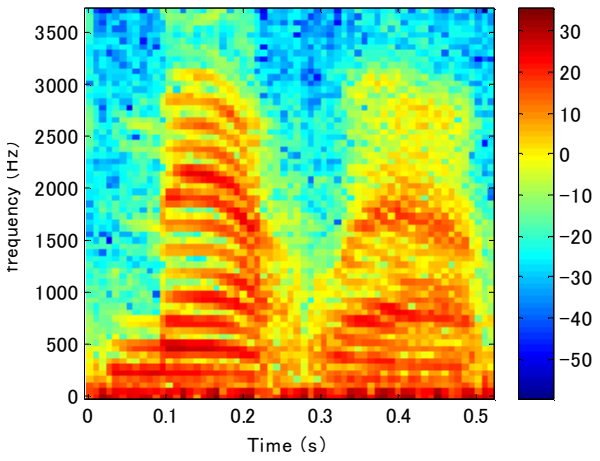
```
>> ex1_2
```



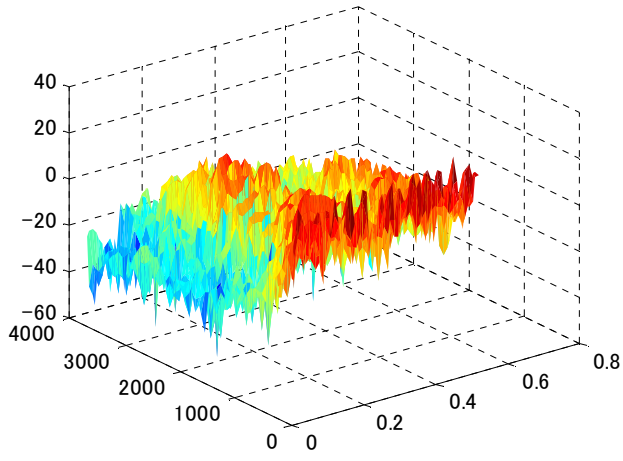
a)　音声信号(時間軸)



b)　Welch 法と Yule-Walker AR 法



c)　スペクトログラム



d)　スペクトログラム(3D)

図 1.2　各種解析手法による計算結果

演習 1.2　音声信号のスペクトル解析 / ex1_2.m　☆ポイント1

```
%　音声信号のスペクトル解析

%入力データ
load mtlb　☆ポイント2                                    %データの読み込み
t=(0:length(mtlb)-1)./Fs;                                %時間軸の設定
plot(t,mtlb), xlim([t(1) t(end)]),grid                   %入力データの表示
xlabel('Time (s)'), ylabel('Magnitude')
title('音声信号')

                                                  ☆ポイント3
[P1,f1]=pwelch(mtlb,hanning(256),128,256,Fs);            %Welch法
[P2,f2]=pyulear(mtlb,14,256,Fs);　☆ポイント4             %YuleWalker法
figure
plot(f1,10*log10(P1),'--',f2,10*log10(P2)),grid          %結果の表示
title('PSD Estimate')
xlabel('Frequency (Hz)')
ylabel('Power Spectrum Density (dB/Hz)')
legend('Welch','Yule-Walker AR')

                                                  ☆ポイント5
[B,f3,t3]=specgram(mtlb,128,Fs,hamming(128),64);         %スペクトログラムの計算
P3=abs(B).^2;
figure
                                                  ☆ポイント6
imagesc(t3,f3,10*log10(P3)),axis xy                      %2次元での表示
xlabel('Time (s)'),ylabel('Frequency (Hz)')
colorbar
figure
surf(t3,f3,10*log10(P3)),shading interp                  %3次元での表示
```

＜解説＞

☆ポイント1

-M-File-

M-FileとはMATLABの関数およびコマンドのステートメントを納めたテキスト形式のファイルのことで、ファイル名は拡張子 **.m** をつけます。M-File はインタプリタ型のプログラムで、実行時にコンパイルやリンクの必要がありません。

M-File は機能によりスクリプトM-Fileとファンクション M-File の2種類の形式に分けられます。スクリプト M-File は、Command Window 上に直接キーボード入力していた関数やコマンドをあらかじめファイル内に記述して、実行させるものです。ファンクション M-File は MATLAB の関数およびコマンドを使用してユーザ定義の新たな外部関数を作成するもので、入力引数と出力引数を伴います。各 Toolbox の多くの関数はファンクション M-File で提供されるので、内部を開いてアルゴリズムを確認したり、変更を加えることができます。

☆ポイント2

- ファイル I/O-

様々なファイル形式のデータの入力、出力を簡単に行うことが可能です。この演習ではMATファイル（MATLAB 固有)を取り込んでいます。

☆ポイント3

Welch法は、信号を分割し、各分割セグメントごとに窓関数をかけてパワースペクトル密度を求め、平均化する手法です。関数 **pwelch** で処理を行うことができます。

```
[Px,freq] = pwelch(sig,window(N),noverlap,nfft,Fs)
```

Px	：パワースペクトル密度	sig	：時間信号ベクトル
freq	：周波数ベクトル	window(N)	：窓関数（N点）　表1.1参照
		noverlap	：オーバーラップ点数
		nfft	：FFT点数
		Fs	：サンプリング周波数

☆ポイント4

Yule-Walker AR 法は、自己回帰（Auto-Regressive, AR）モデルの係数をユール・ウォーカ方程式を解くことにより求め、その周波数特性からスペクトル推定を行う手法です。関数 **pyulear** で処理を行うことができます。

```
[Px,freq] = pyulear(sig,ord,nfft,Fs)
```

Px	：パワースペクトル密度	sig	：時間信号ベクトル
freq	：周波数ベクトル	ord	：ARモデルの次数
		nfft	：FFT点数
		Fs	：サンプリング周波数

☆ポイント5

スペクトログラムは、セグメントごとに分割した信号に窓関数をかけてFFTを行い、各セグメントの結果を時間軸方向に並べたもので、周波数スペクトルの時間変化を観測できます。関数**specgram**で処理を行うことができます。

```
[B,freq,time] = specgram(sig,nfft,Fs,window(N),noverlap);
```

B	：スペクトログラム結果	sig	：時間信号ベクトル
freq	：周波数ベクトル	nfft	：FFT点数
time	：時間ベクトル	Fs	：サンプリング周波数
		window(N)	：窓関数（N点）表1.1参照
		noverlap	：オーバーラップ点数

☆ポイント6

関数**imagesc**は、カラーマップの全領域にイメージデータをスケールリングし、イメージを表示しする関数で、ここではスペクトログラムの結果を2次元で表示します。関数**surf**は、3次元サーフェスプロットを行う関数で、ここではスペクトログラムの結果を3次元で表示します。

＜補足＞

- 音声データの取得 -

Data Acquisition Toolboxを利用することで、Windows サウンドカード（もしくは、Agilent Technology、National Instruments社等のデータ収集用ハードウエア）からデータを収集することもできます。下記コマンドを実行すると、サウンドカードから収集したアナログ信号に対して逐一パワースペクトルを計算している様子を確認することができます。

```
>> demoai_fft                    %入力信号に対するパワースペクトル
```

- 音声データの出力 -

MATLABでは、音声データの入出力を行うことができます。以下のようにCommand Window上に入力すると、演習1.2で用いた入力データの音声を出力することができます。(PCにサウンド機能が付いている場合のみ)

```
>> sound(mtlb,Fs)              %音声信号の再生
```

＜参考＞

Signal Processing Toolbox には、窓関数を設計・表示する GUI ツールとして、Windows Design & Analysis Tool(WINTool)が用意されています。Command Window 上で

```
>> wintool
```

と入力すると WINTool を立ち上げることができます。

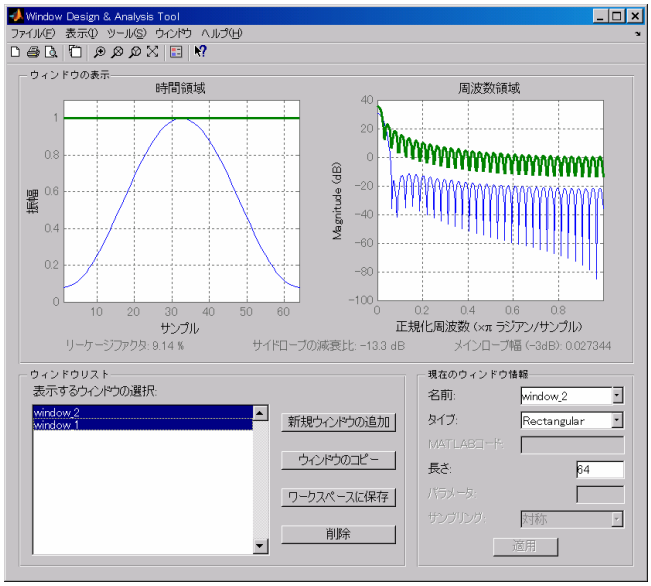


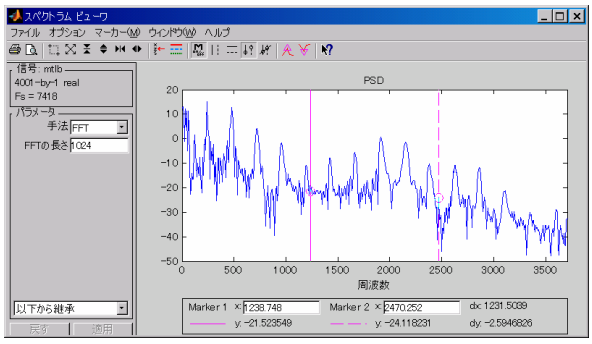
図 1.3　Windows Design & Analysis Tool(WINTool)

また、信号、フィルタ、スペクトルの読み込みや、解析を行う GUI として、SPTool が用意されています。SPTool には、各種スペクトル推定法を GUI 上で行うことのできる「スペクトラムビューワ」が提供されています。Command Window 上で

```
>> sptool
```

と入力すると SPTool を立ち上げることができます。

a)　SPTool



b)　スペクトラムビューワ

図 1.4　SPTool - スペクトラムビューワ

第2章 フィルタ設計

ディジタルフィルタは非常に幅広い分野で利用されていて、その目的も様々であるため、数多くの設計方法があります。Signal Processing Toolbox、Filter Design Toolbox には各種ディジタルフィルタの設計・解析関数が用意されているため、様々な設計手法を用いることができます(表 2.1)。ここでは、MATLAB によるフィルタ設計やフィルタ特性の検討、テスト信号に対するシミュレーションの一連の流れを紹介をします。

表 2.1　ディジタルフィルタ設計関数

Signal Processing Toolbox	
butter	Butterworth フィルタの設計
cheby1	Chebyshev Ⅰ型フィルタの設計
cheby2	Chebyshev Ⅱ型フィルタの設計
ellip	楕円フィルタの設計
yulewalk	Yulewalk 巡回型フィルタの設計
cfirpm	複素数かつ非線形位相の等リップル FIR フィルタの設計
fir1	窓関数法 FIR フィルタの設計（標準応答）
fir2	窓関数法 FIR フィルタの設計（任意応答）
firrcos	コサインロールオフフィルタの設計
firpm	Parks-McClellan 最適 FIR フィルタの設計
Filter Design Toolbox	
firlpnorm	最小Pノルム最適FIRフィルタ設計
firgr	一般REMEZ FIRフィルタ設計
iirgrpdelay	群遅延を与えてオールパスフィルタを設計
iirlpnorm	最小Pノルム最適IIRフィルタ設計
iirlpnormc	制約付最小PノルムIIRフィルタ設計

<MATLAB におけるディジタルフィルタの表現 >

一般にディジタルフィルタは Z 変換により、式 2.1 の離散伝達関数で表現できます。

$$H(z)=\frac{\sum_{k=0}^{N} b_k z^{-k}}{1+\sum_{k=1}^{N} a_k z^{-k}}=\frac{b_0+b_1 z^{-1}+b_2 z^{-2}\cdots+b_{N-1} z^{-(N-1)}+b_N z^{-N}}{1+a_1 z^{-1}+a_2 z^{-2}\cdots+a_{N-1} z^{-(N-1)}+a_N z^{-N}} \qquad \text{式 2.1}$$

MATLAB では、ディジタルフィルタを、式 2.1 の有理関数で表される分子と分母の多項式係数のベクトルで表現します。

例として、式 2.2 で表される 2 次のディジタルフィルタを MATLAB 上で定義します。

$$H(z)=\frac{2-3z^{-1}+5z^{-2}}{1+4z^{-1}-7z^{-2}} \qquad \text{式 2.2}$$

フィルタの分子と分母の係数ベクトルの変数をそれぞれ b,a として、

```
>> b = [2 -3 5];
>> a = [1 4 -7];
```

となります。

2.1 窓関数法によるFIRフィルタの設計

MATLAB による簡単なフィルタ設計の例として、カットオフ周波数 0.5(ナイキスト周波数を 1 とする)、20 次の FIR ローパスフィルタの設計を窓関数法を用いて行い、その振幅応答と位相応答の表示を行います。

演習 2.1　窓関数法による FIR ローパスフィルタの設計

```
>> b=fir1(20,0.5);☆ポイント1    %インパルス応答の係数計算（フィルタの設計）
>> freqz(b,1) ☆ポイント2         %周波数応答の計算＆プロット
```

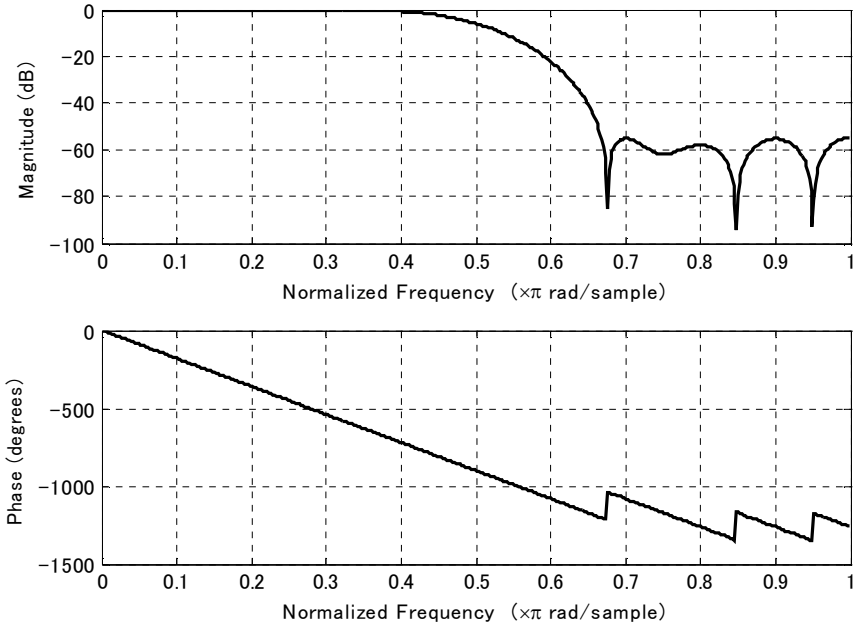


図 2.1　FIR ローパスフィルタの振幅応答・位相応答

＜解説＞

☆ポイント 1

窓関数法は、有限インパルス応答に窓関数を適用したフィルタの設計手法で、関数**fir1**によりFIRフィルタの設計を行うことができます。デフォルトではHammingウィンドウを適用したローパスフィルタを算出しますが、適用する窓関数の種類、通過帯域の変更を行うことができます。

```
b=fir1(n,Wn,window)
```

b　　：フィルタの分子係数ベクトル

n　　：フィルタ次数
Wn　　：カットオフ周波数
window　：窓関数　表1.1参照

☆ポイント2

関数 **freqz** を用いることにより、設計したフィルタの振幅応答・位相応答を簡単に計算、表示することができます。

```
freqz(b,a)
```

b　：フィルタの分子係数ベクトル
a　：フィルタの分母係数ベクトル

＜補足＞

この演習と同様のプログラムは ml2_1.m に用意されています。プログラムを参照するには、Command Window 上に以下のように入力してください。

```
>> edit ml2_1
```

Command Window 上に以下のように入力すると、演習 2.1 と同様の結果が得られます。

```
>> ml2_1
```

2.2 各種IIRフィルタの設計

ここでは、以下の仕様を持つバンドパス型の Yulewalk,Chebyshev Type 1, Ellipse,Butterworth フィルタをプログラム環境で設計し、その応答を表示し各フィルタの比較を行います(図 2.2)。

仕様：

・サンプリング周波数　　: 1000[Hz]
・通過帯域　　: 100 ～ 200[Hz]
・遮断帯域　　: 50 ～ 250[Hz]
・通過帯域リップル　　: 0.3[dB]
・遮断特性　　: 30[dB]

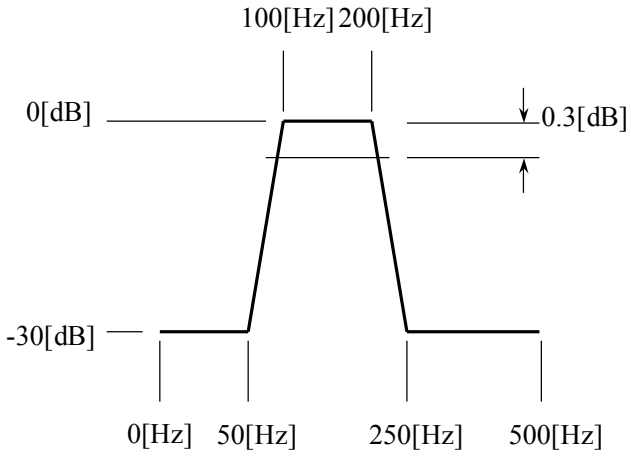


図 2.2　フィルタの仕様

演習 2.2　各種 IIR フィルタの設計 / ex2_2.m(一部省略)

```
%　各種IIRディジタルフィルタの設計
Wp=pp/(0.5*Fs);                          % 通過帯域、ナイキスト周波数で正規化
Ws=ss/(0.5*Fs);                          % 遮断帯域、ナイキスト周波数で正規化

% YuleWalk Filter
ff=[0 Wp(1) Wp(1) Wp(2) Wp(2) 1];        % 周波数ベクトルの設定
mm=[0 0 1 1 0 0];                        % フィルタの振幅の設定
[yb,ya]=yulewalk(10,ff,mm);☆ポイント1    % フィルタ係数の算出
yh=freqz(yb,ya,512);                     % フィルタの周波数応答の算出
yh(find(yh==0)) = eps;                   % ゼロ割の防止
yh_a=20*log10(abs(yh));

% Chebyshev Type 1 Filter ☆ポイント2
[n1,w1]=cheb1ord(Wp,Ws,Rp,Rs);           % 最小次数の算出
[c1b,c1a]=cheby1(n1,Rp,w1);☆ポイント2    % フィルタ係数の算出
c1h=freqz(c1b,c1a,512);                  % フィルタの周波数応答の算出
c1h(find(c1h==0)) = eps;                 % ゼロ割の防止
c1h_a=20*log10(abs(c1h));

% Ellipse Filter
[n2,w2]=ellipord(Wp,Ws,Rp,Rs);           % 最小次数の算出
[eb,ea]=ellip(n2,Rp,Rs,w2);              % フィルタ係数の算出
eh=freqz(eb,ea,512);                     % フィルタの周波数応答の算出
eh(find(eh==0)) = eps;                   % ゼロ割の防止
eh_a=20*log10(abs(eh));

% Butterworth Filter
[n3,w3]=buttord(Wp,Ws,Rp,Rs);            % 最小次数の算出
[bb,ba]=butter(n3,w3);                   % フィルタ係数の算出
bh=freqz(bb,ba,512);                     % フィルタの周波数応答の算出
bh(find(bh==0)) = eps;                   % ゼロ割の防止
bh_a=20*log10(abs(bh));

frq=(0:511)/1024*1000;                   %周波数帯域
plot(frq,[yh_a c1h_a eh_a bh_a])         % 結果の表示
axis([0 500 -100 5]),grid
title('Filter Response')
xlabel('Frequency [Hz]'),ylabel('Gain [dB]')
legend('YuleWalk Filter','Chebyshev type 1',...
           'Ellipse filter','Butterworth filter')
```

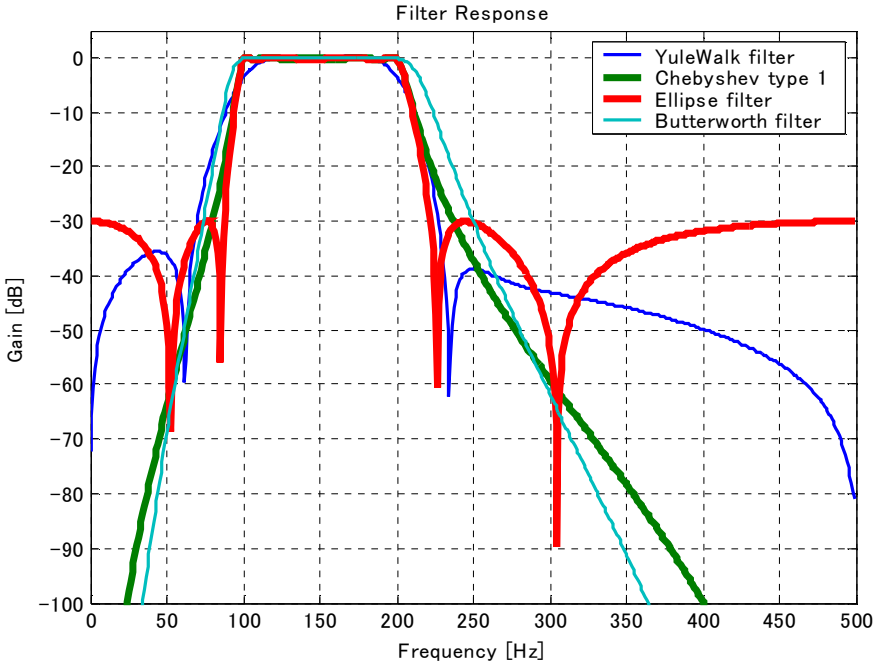


図 2.3　各種 IIR フィルタの設計結果

＜解説＞

☆ポイント 1

YuleWalk 巡回型フィルタの設計関数 **yulewalk** では、フィルタの仕様を周波数ベクトルとそれに対応する振幅ベクトルで与えます。周波数はナイキスト周波数を 1 として正規化されています。

```
[yb,ya] = yulewalk(n,ff,mm)
```

yb	：IIR フィルタの分子係数ベクトル	n	：フィルタの次数
ya	：IIR フィルタの分母係数ベクトル	ff	：周波数ベクトルの仕様
		mm	：フィルタの振幅の仕様

☆ポイント 2

多くのフィルタ設計関数は、フィルタの仕様を与えたとき、その仕様を満たす最小の次数を選択する関数があり、その結果を利用して簡潔にフィルタ設計ができます。

```
[n1,w1] = cheb1ord(Wp,Ws,Rp,Rs)
```

n1	：仕様を満たすフィルタの最小次数	Wp	：通過帯域のエッジ周波数
w1	：設計時に与える周波数	Ws	：遮断帯域のエッジ周波数
		Rp	：通過帯域のリップル
		Rs	：遮断特性

```
[c1b,c1a] = cheby1(n1,Rp,w1);
```

c1b	：IIR フィルタの分子係数ベクトル	n1	：フィルタの次数
c1a	：IIR フィルタの分母係数ベクトル	Rp	：通過帯域のリップル
		W1	：設計周波数

＜発展＞

Simulinkでは、システムをブロック線図の形で記述し、入力信号に対する応答をシミュレーションにより即座に観測することができます。また、MATLAB環境で設計したフィルタ係数を取り込むことができます。その際、変数を加工する必要はありません。ここでは、演習2.2で設計した各フィルタの係数をSimulink環境に取り込みシミュレーションを行います。

Command Window上に以下のように入力し、図2.4のモデルを起動します。

```
>> adpf2_2
```

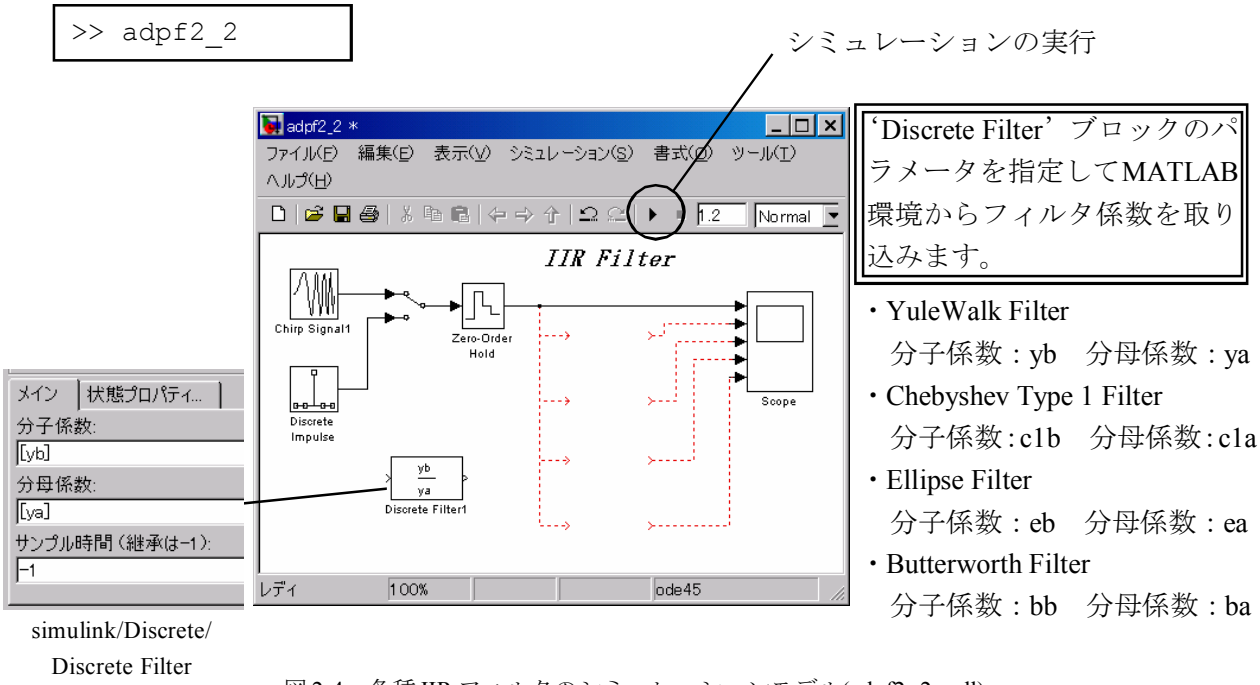
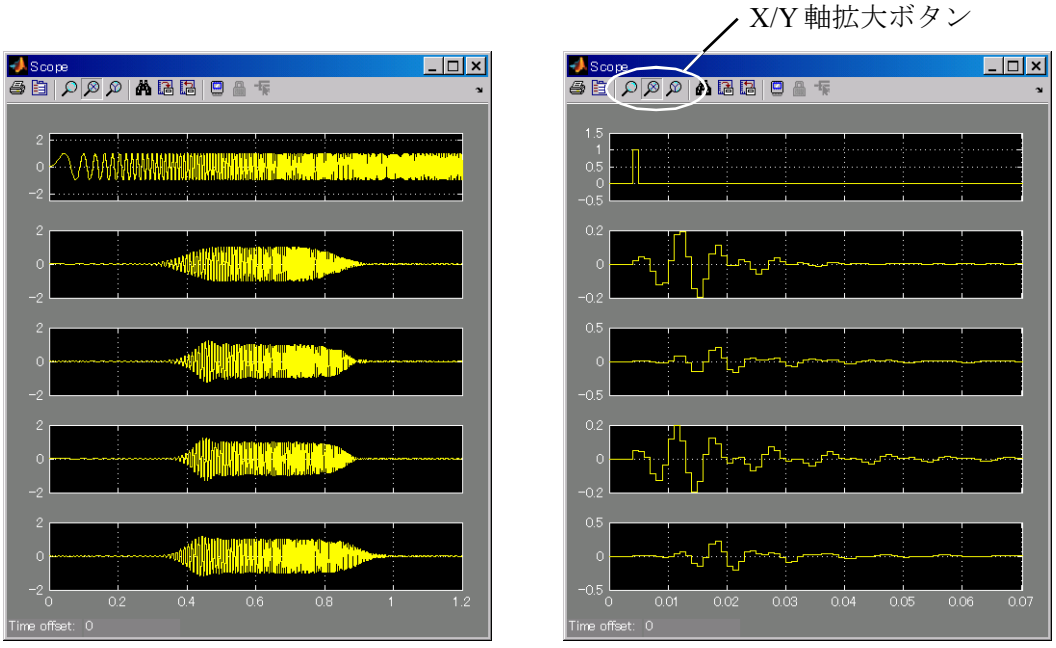


図2.4　各種IIRフィルタのシミュレーションモデル(adpf2_2.mdl)

以上のモデル作成が終了しましたら、Switchブロックをダブルクリックして入力信号を切り換えてシミュレーションを実行します。下記はその結果です。



a) 入力＝チャープ信号　　b) 入力＝インパルス信号

図2.5　シミュレーション結果

＜補足 2＞

同様のモデルはsl2_2.mdlに用意されています。Command Window上に以下のように入力すると、図2.4のモデルが表示されます。

```
>> sl2_2
```

＜参考＞

Signal Processing Toolbox、Filter Design Toolbox、Signal Processing Blockset共通のディジタルフィルタ設計GUIとして、Filter Design & Analysis Tool(FDATool)が用意されています。FDAToolはフィルタ設計機能の一部をGUI上で行いフィルタ設計機能を強力にサポートします。MATLABコマンドプロンプトで

```
>> fdatool
```

と入力すると、FDAToolを立ち上げることができます。

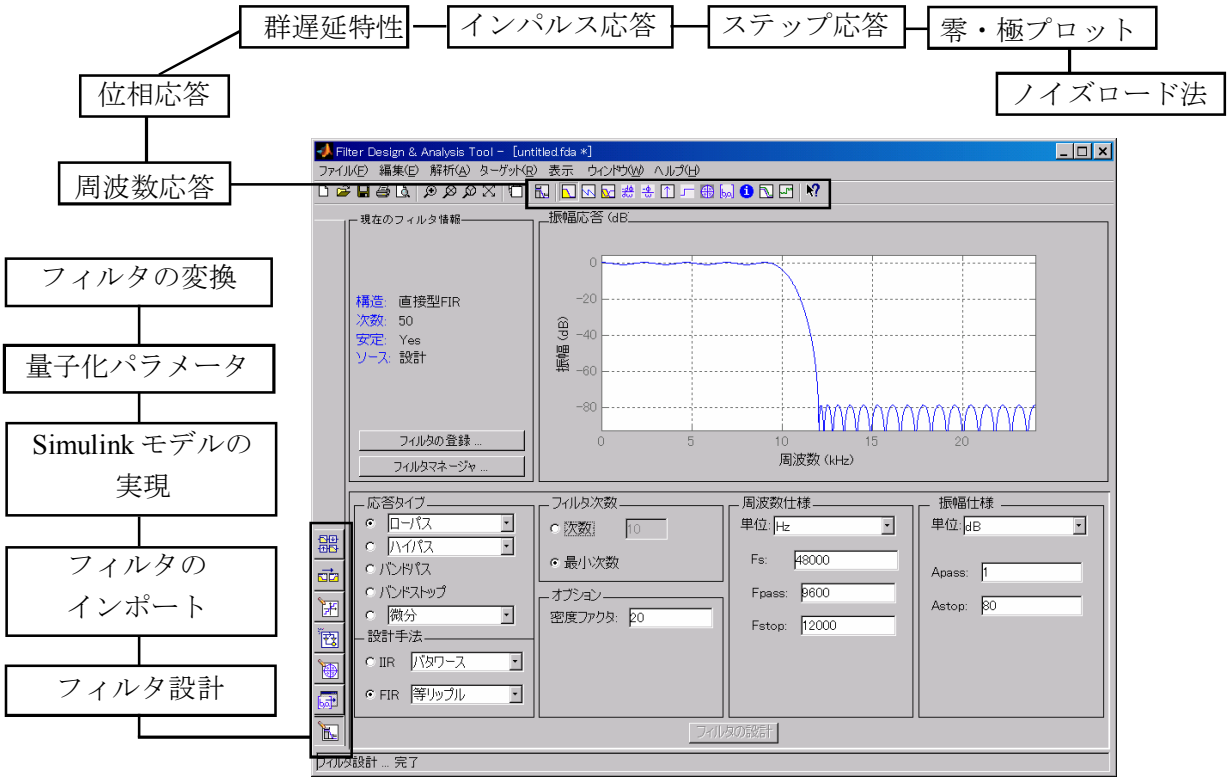


図2.6　Filter Design & Analysis Tool(FDATool)

<FDATool の主な機能＞

【フィルタ設計機能】

フィルタ設計機能の一部をGUI上で行うことができます。FDAToolで設計したフィルタをそのままSimulink環境で利用できます。

【Simulink モデルの実現】

FDATool で設計したフィルタを、Simullinkブロックに変換することができます。

【量子化パラメータ】

フィルタの量子化を行う際のパラメータ（ビット長、スケーリングなど）を指定できます。

【振幅・位相・群遅延・インパルス・ステップ応答、零・極プロット】

一般的なフィルタ解析の手法を利用できます。

【ノイズロード法】

量子化フィルタの周波数応答を計算する方法としてノイズロード法を利用できます。モンテカルロトライアルを使ってノイズライクな信号をフィルタに入力し、フィルタの周波数応答を計算します。

2.3 櫛形フィルタ

図2.7に示す構造をもつフィルタは櫛形フィルタ、またはコム（Comb）フィルタと呼ばれます。このフィルタはその振幅の形状から櫛形フィルタと呼ばれ、2/N周期の周波数で減衰量が-∞になるという特徴があります。これをブロック線図環境であるSimulink上でモデル化し、シミュレーションを行います。

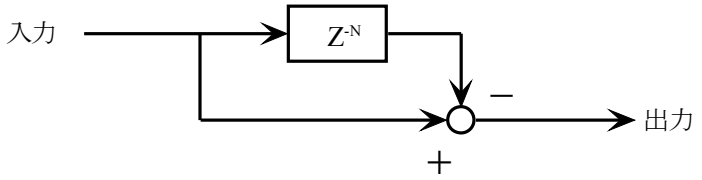


図2.7　櫛形フィルタの構造

z^{-N} は N サンプル信号を遅らせることを意味します。次数を $N=8$ とすると、図2.7に示す櫛形フィルタの伝達関数 $H(z)$ は、式2.3のようになります。

$$H(z)=1-z^{-8} \qquad \text{式2.3}$$

8次の櫛形フィルタを設計し、その振幅特性と位相特性表示します(図2.8)。

```
>> b = [1 zeros(1,7) -1];          %b=[1 0 0 0 0 0 0 0 -1]と等しい
>>freqz(b,1,1024)
```

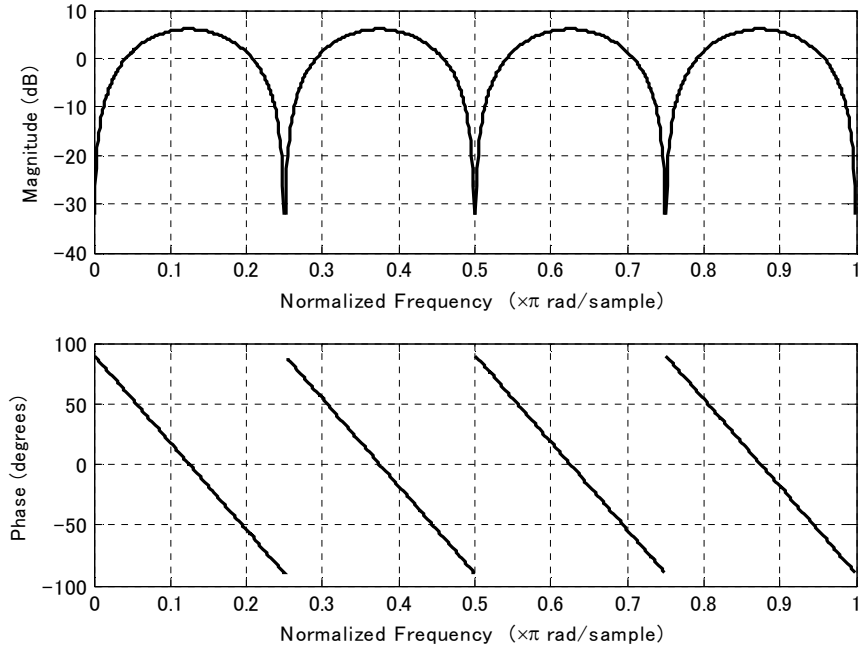
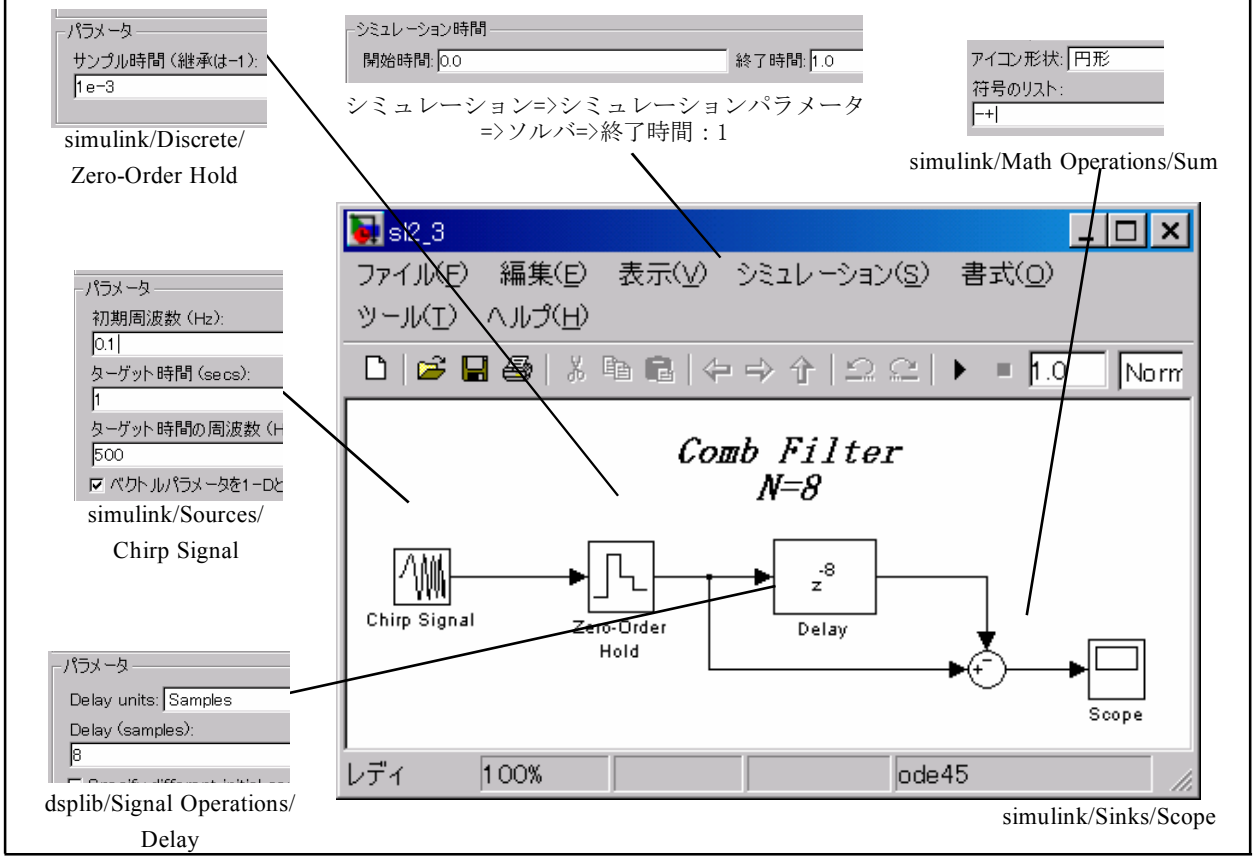


図2.8　8次櫛形フィルタの振幅応答・位相応答

8次の櫛形フィルタをSimulinkでモデル化します。テスト信号としてスイープサイン波形を入力し、櫛形の周波数特性を確認します。このとき、サンプリング周波数は1/1000[s]とします。

演習 2.3　櫛形フィルタのモデル / sl2_3.mdl



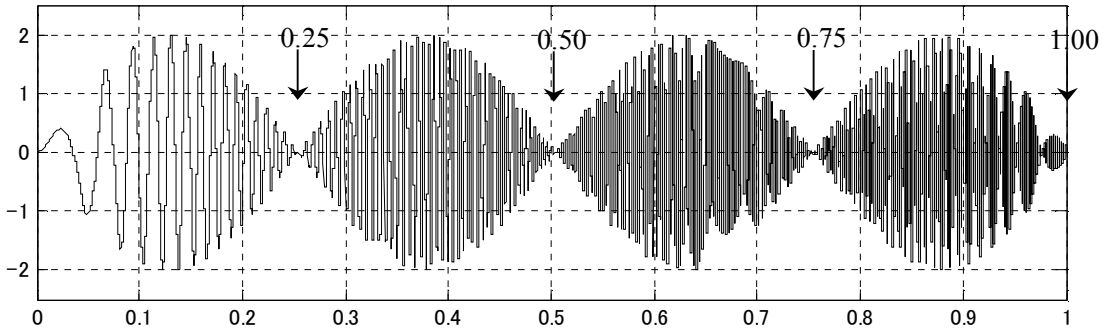


図 2.9　シミュレーション結果

<補足>

この演習と同様のモデルは sl2_3.mdl に用意されています。Command Window 上に以下のように入力するとモデルが表示されます。

```
>> sl2_3
```

第3章 アプリケーション例題

ここでは、下記のデモモデルを例として取り上げ、Simulink の利点について確認します。

1: 時間 - 周波数解析システム
2: 動画像のエッジ検出システム

3.1 Simulinkによる時間-周波数解析システム

時間に伴い信号に含まれる周波数成分が変化する非定常な音声信号に対しては、一般に時間-周波数解析が可能な短時間 FFT（Short-Time FFT）が用いられます。短時間 FFT は、窓関数により切り出した一定期間の信号毎に FFT を実行し、スペクトルの時間変化を推定する手法です。Signal Processing Blockset は短時間フーリエ変換を含むパワースペクトル推定関数や、フィルタ設計、数学、統計処理関数などディジタル信号処理システム構築に必要な多くのブロックが提供されています。ここでは外部から入力した音声信号に対して時間 - 周波数解析を行った例を紹介します。Command Window 上に以下のように入力するとモデルが表示されます。

```
>> dspsfft1
```

3.1.1 Simulink モデル構成

Simulink によるモデリング例を図 3.1 に示します。オーディオデバイスから取り込んだ実時間の音声信号を Simulink 上に定義します（From Wave Device）。Buffer1 ブロックにより 1 フレーム当たり 128 サンプルの信号を作成し、その後連続的に高速フーリエ変換を行います。Short-Time FFT ブロック内部では、一定時間毎に切り出した信号に対して窓関数を適用した後、FFT を実行します。次に計算結果の 2 乗を平均化し、窓関数の 2 乗和で正規化した信号が Short-Time FFT ブロックの出力となります。最後に Buffer2 ブロックにより周波数情報をもつ信号を時間軸に並べ、Matrix Viewer ブロックにより 3 次元の情報をもった信号を表示させます。

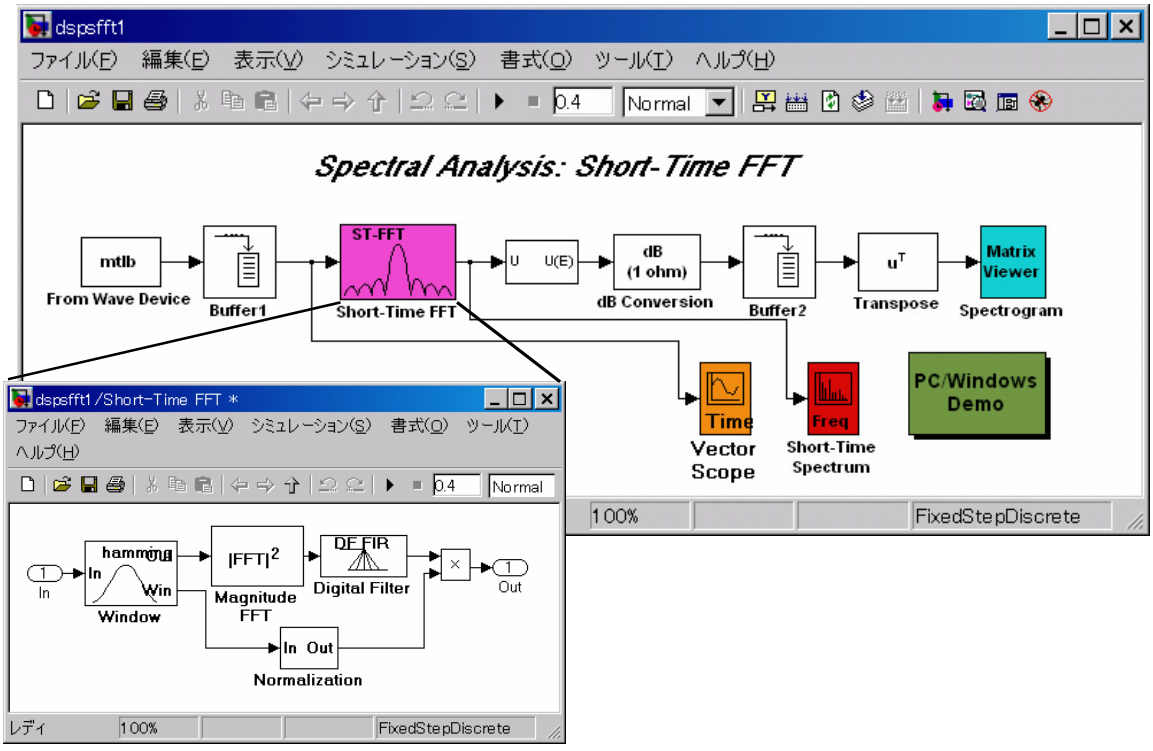
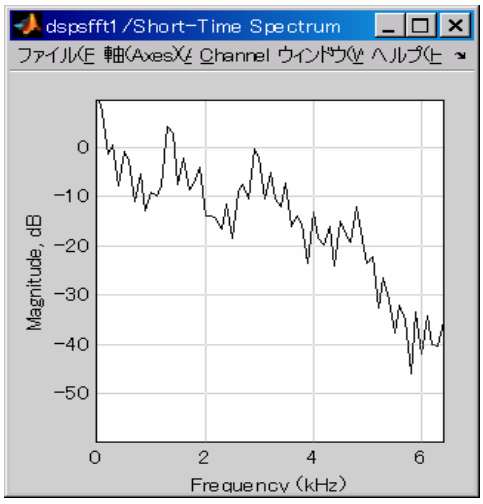


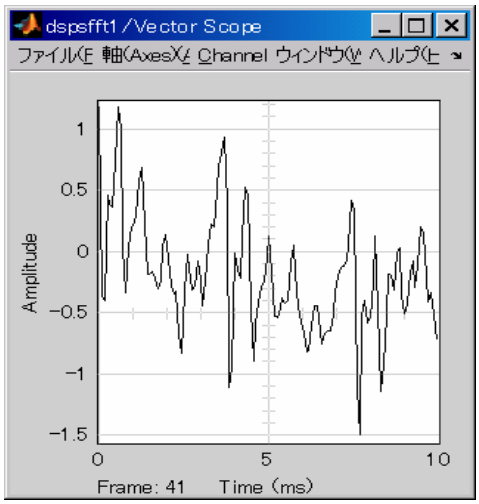
図 3.1　時間 - 周波数解析システム

3.1.2 シミュレーション結果

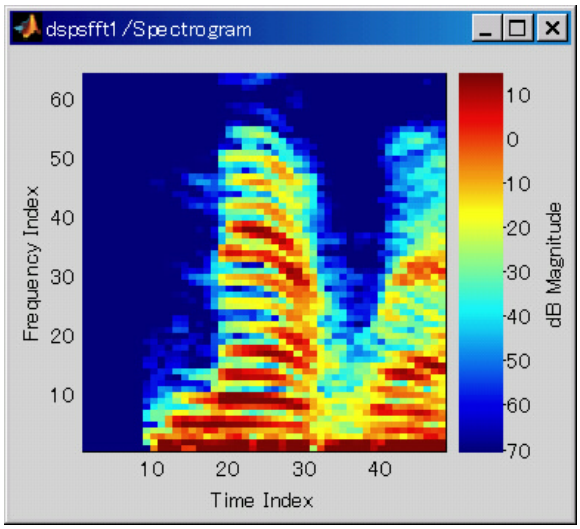
シミュレーション結果を図 3.2 に示します。(a)はスペクトログラムです。横軸は時間、縦軸は周波数レベルを示し、色によりスペクトル強度を表しています。(b)は一定時間毎に切り出した信号の周波数スペクトルで、(c) は音声信号の時間応答です。なお、Matrix Viewer ブロックを Simulink システム上に配置するだけで、時間で変化する周波数スペクトルを連続的に確認することができます。



(b)　周波数スペクトル



(c)　音声信号の時間応答



(a)　音声信号のスペクトログラム

図 3.2　シミュレーション結果

3.1.3 まとめ

Simulink /Signal Processing Blockset を用いて音声信号の時間 - 周波数解析および可視化を行いました。Simulink を使用して、短区間におけるスペクトル強度を連続的に計算させた結果をスペクトログラムに表示することで音声信号の特徴を簡単に観測することができます。

3.2 Simulinkによる動画像のエッジ検出システム

MATLAB/Simulink では、Fortran/C 言語で作成されたプログラムとのインタフェース機能が提供されています。これらの機能を利用することで、Fortran/C言語で記述されたプログラム資産を取り込んで解析・検証を行うことができるため、既存のプログラム資産を有効に活用できます。ここでは、動画像のエッジ検出システムを例にとり、CプログラムをSimulinkのブロックとして取り込むことを可能にするS-Function機能について簡潔に説明します。Command Window上に以下のように入力するとモデルが表示されます。

```
>> vid_edge
```

3.2.1 Simulink モデル構成

図3.3にシステム全体を示します。まず、Signal From Workspaceブロックでは、時間的に連続した120行160列の画像を285フレーム取り込みます（画素の階調：0～255)。なお、図3.3に記載されているブロック間のラインを見ると次の記述がされています。 “uint8[120 × 160]”。これは、1サンプルあたりのデータが120行160列の符号なし8bit整数であることを示しています。このようにSimulinkでは、スカラ信号ばかりでなく、2次元信号を扱うことができるため、2次元信号を扱う画像処理システム等を直感的に作成することができます。

次に、Edge Detectionブロックでは、水平・垂直方向の成分がよく検出されるといわれるPrewitt法によるエッジ検出を行います。なお、この計算法はImage Processing Toolboxのedge関数でサポートされていますが、同様の処理をCコードで実現します。CコードをSimulinkブロックに取り込むためには、S-Function機能を利用します（図3.4)。S-Functionを記述する方法は2つあり、実現内容に応じて選択することができます（テンプレートファイルの利用 /S-Function Builder の利用)。

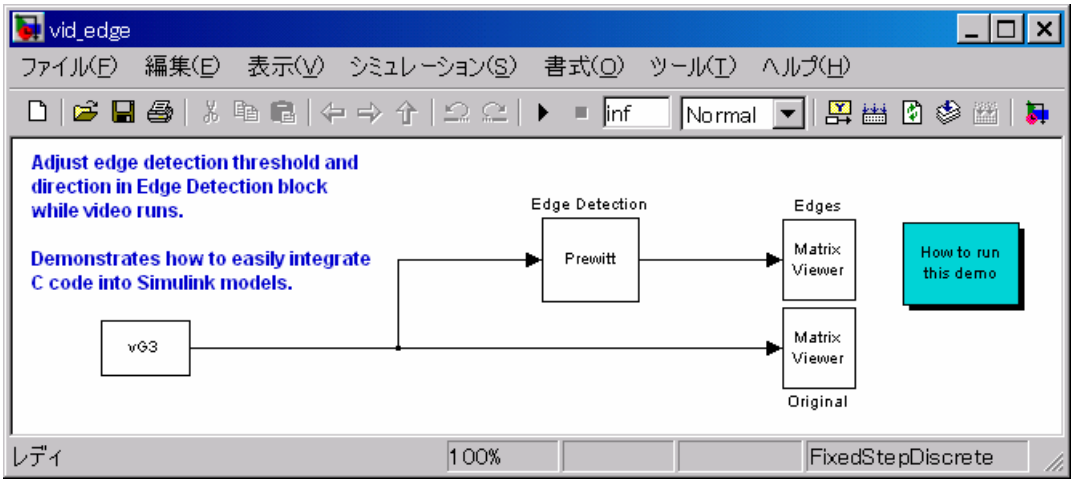


図3.3　動画像のエッジ検出システム

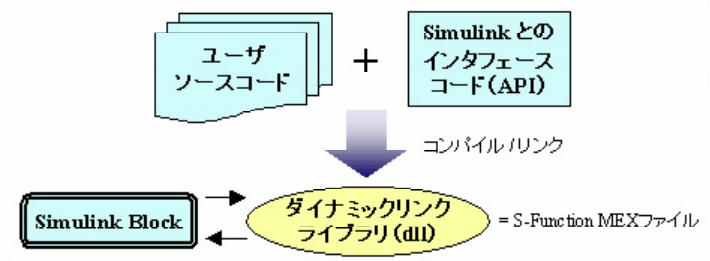


図3.4　S-Function の仕組み

テンプレートファイルで実現する方法を利用する場合、各種設定コードを編集する必要がありますが、S-Function Builder を利用するとシステムの初期設定や C コードのコーディングを GUI 上で行うことができるだけでなく、ビルドボタンを押すだけで C コードをビルドして S-Function の MEX ファイル（dll）を自動生成することができます（図 3.6）。これにより、各種設定のコーディングの手間を省くことができます。Edge Detection ブロックを右クリックし、「マスクブロックのモデル表示」を選択すると、S-Function Builder ブロックで構築された Edge Detection モデルが表示されます（図 3.5）。

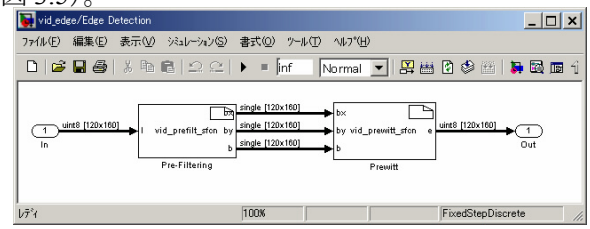


図 3.5　Edge Detection モデル

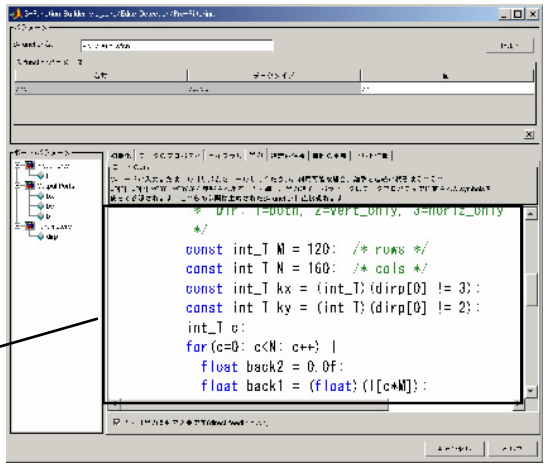


図 3.6　S-Function Builder Block の GUI

3.2.2 シミュレーション結果

Edge Detection ブロックには、パラメータとしてスレッシュホールド値（=Cutoff）とエッジ検出の方向（=Derection）が指定されています（図 3.7）。このパラメータをシミュレーション中に変化させて、結果を逐一確認することができます （図 3.8）。

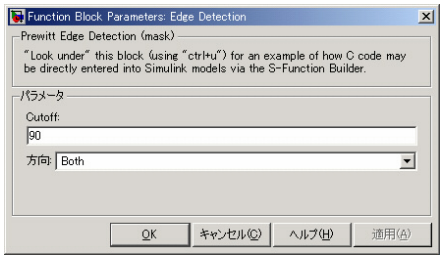


図 3.7　Edge Detection ブロックのパラメータウィンドウ

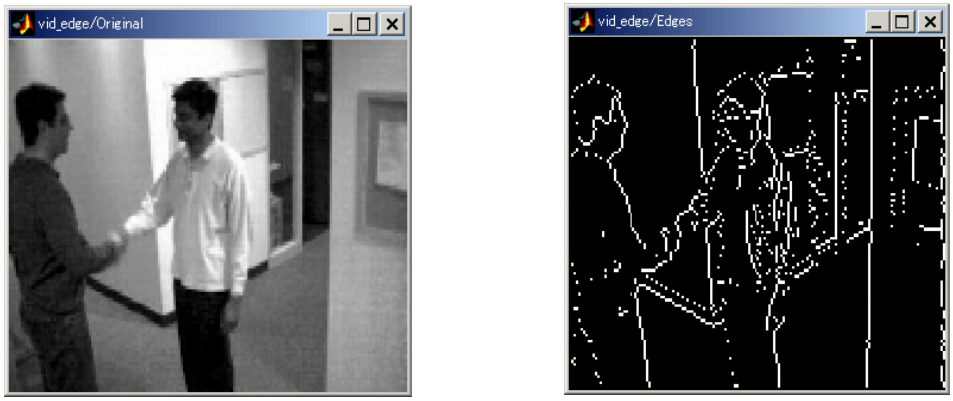


図 3.8　シミュレーション結果

3.2.3 まとめ

Simulinkによって動画像処理システムを直感的に実現でき、パラメータ変更に伴う結果を即座に確認することができます。また、S-Function 機能により容易に C コードを Simulink のブロックとして取り込むことができることを紹介しました。

3.3 デモンストレーション

Signal Processing Blockset では基本的な処理から大規模アプリケーションまで、多数のリファレンスモデルが提供されております。これにより、ユーザシステムの早期構築を支援します。

以下のコマンドでデモンストレーション一覧を起動することができます

```
>> demo
```