CONTENTS

ビジュアルヘルプ – FilterFIR 2

FilterFIR コマンドのヘルプ 2

ビジュアルヘルプ – FilterFIR

FilterFIR コマンドは、メニュー Analysis→Filter を選択して、ダイアログでも操作できます。
ここではダイアログと対比して説明します。

FilterFIR コマンドのヘルプ

FilterFIR [/COEF[=*coefsWaveName*] /DIM=*d* /E=*endEffect* /ENDV={*sv*[,*ev*]} /HI={*f1*, *f2*, *n*} /LO={*f1*, *f2*, *n*} /NMF={*fc*, *fw* [, *eps*, *nMult*]} /WINF=*windowKindName*] *waveName* [, *waveName*]...

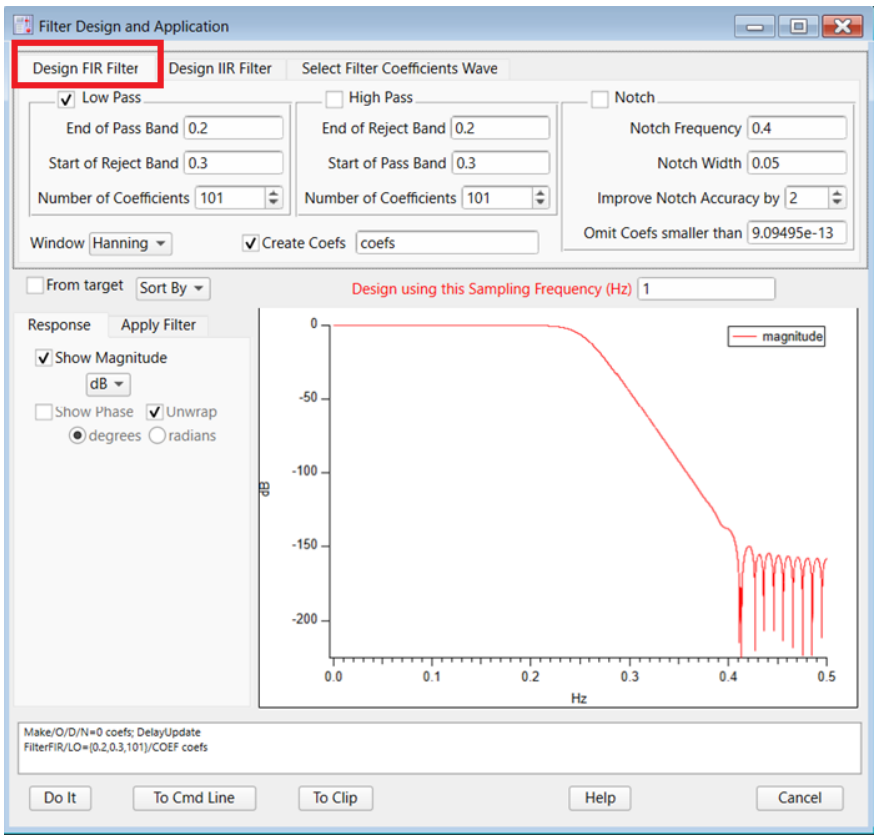
FilterFIR コマンドでは、各 *waveName* に対して、自動的に設計されたフィルター係数、あるいは *coefsWaveName* を使って、時間領域の手法により畳み込み演算を行います。

自動的に設計されたフィルター係数は、単純なウィンドウベースのローパスフィルターとハイパスフィルター、あるいは最大平坦性のノッチフィルターです。
複数のフィルター設計を組み合わせて、複合フィルターを作成します。
このフィルターは、オプションで最初の *waveName* に配置することも、単に *waveName* 内のデータをフィルタリングするために使うこともできます。

waveName 内のデータポイント数に比べてフィルタ係数の値がはるかに少ない場合、FilterFIR は Convolve よりも高速にデータをフィルタリングします。

注記

FilterFIR は、廃止された SmoothCustom コマンドに代わるものです。



パラメーター

waveName は、それ自身とフィルターの畳み込みによって上書きされる宛先ウェーブです。

waveName は多次元データである場合がありますが、/DIM で指定された次元のみがフィルタリングの対象となります（２次元のフィルタリングについては、MatrixFilter コマンドを参照してください）。

waveName が複素数の場合、実部と虚部はそれぞれ個別にフィルタリングされます。

フラグ

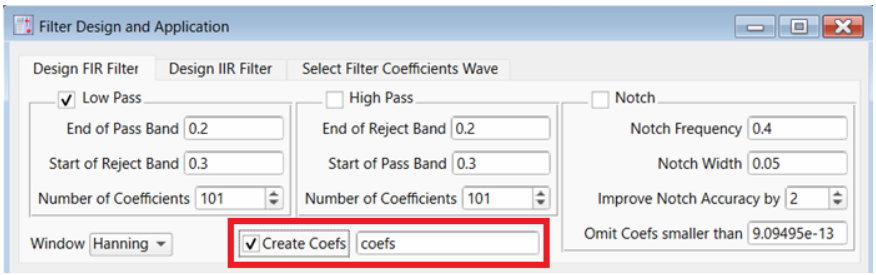
/COEF [=*coefsWaveName*]

最初の出力 *waveName* を、フィルタリング後の結果ではなくフィルター係数に置き換えます。また、*coefsWaveName* が指定されている場合は、出力ウェーブを *waveName* と *coefsWaveName* 内の係数との畳み込み結果に置き換えます。

coefsWaveName は、宛先の *waveNames* のいずれでもあってはなりません。単精度または倍精度の数値であり、１次元でなければなりません。

出力が入力に対してずれるのを防ぐため、*coefsWaveName* の要素数は奇数でなければならず、「center」係数はウェーブの真ん中に配置する必要があります。

係数は通常、中点を軸に対称になりますが、FilterFIR ではこれを強制しません。



/DIM=*d*　　フィルタリングするウェーブの次元を指定します。

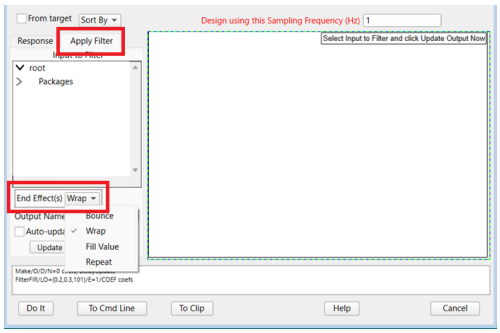
d=-1：　　ウェーブ全体を 1D として扱う（デフォルト）
d>=0：　　行や列などに沿って処理を行う

/DIM=0 を使うと、多次元ウェーブデータ（各行が特定の時点におけるすべての音声サンプルで構成される）の個々の列（それぞれがチャネル、例えば左チャネルや右チャネルなど）にフィルターを適用できます。

/E=*endEffect*　　欠落している隣接値を補完する時に、ウェーブ（"w"）の両端をどのように扱うかを決定します。*endEffect* の取り得る値は以下の通りです。

0：　バウンス法（デフォルト）。欠落している w[-i] の代わりに w[i] を、欠落している w[n+i] の代わりに w[n-i] を使う。

1：　ラップ法。欠落している w[-i] の代わりに w[n-i] を使い、その逆も同様に行う。



2：　0 で埋める。/ENDV={0} と同じ。

3：　フィル法。欠損値 w[-i] の代わりに w[0] を、欠損値 w[n+i] の代わりに w[n] を使う。

/ENDV={*sv*[, *ev*]}

各フィルタリングされたウェーブについて欠損している隣接値を補完する時、データの開始点より前の欠損値は *sv* で置き換えられます。

データの末尾以降の欠損値は、指定がある場合は *ev* で、*ev* が省略された場合は *sv* で置き換えられます。/ENDV を指定すると、/E=2 と同じ意味になります。

/ENDV は Igor Pro 9.0 で追加されました。

/HI={*f1*, *f2*, *n*}　/WINF で別のウィンドウが指定されていない限り、Hanning ウィンドウを使って、ウィンドウ法に基づくハイパスフィルターを作成します。

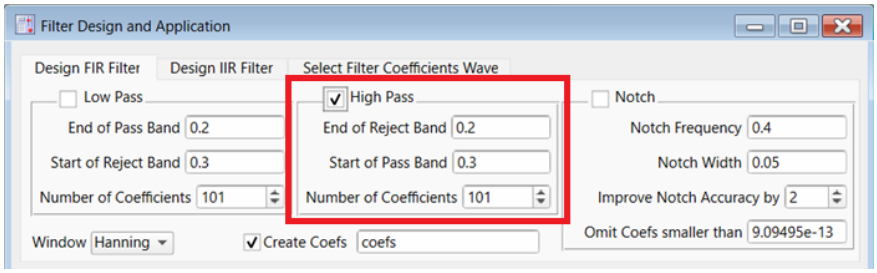
f1 と *f2* は、サンプリング周波数の分数で表されるフィルターの設計周波数であり、0.5（正規化ナイキスト周波数）を超えてはなりません。

f1 はリジェクト帯域の終端であり、*f2* はパス帯域の始点です。

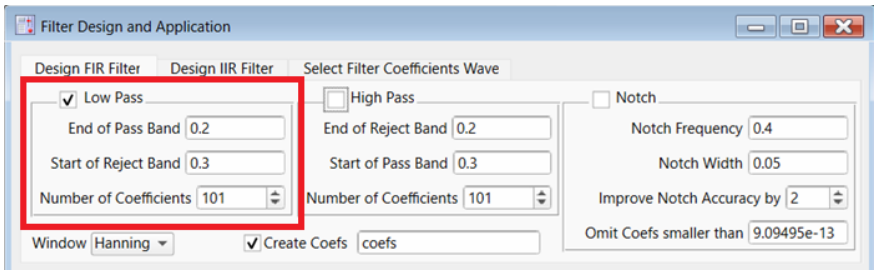
0 < f1 < f2 < 0.5

n は、生成する FIR フィルターの係数の数です。この数値が大きいほど、ストップバンドの遮断性能が向上します。最初の設定値としては 101 が適しています。

/HI と /LO の両方を指定して、バンドパスフィルターを作成します。



/LO={*f1*, *f2*, n}　ローパスフィルターを作成します。*f1* は通過帯域の終了周波数、*f2* は遮断帯域の開始周波数、*n* は FIR フィルターの係数の数です。詳細は、/HI を参照してください。



/NMF={*fc*, *fw* [, *eps*, *nMult*]}

fc を中心とし、-3dB 帯域幅が *fw* の最大平坦フィルターを作成します。*fc* と *fw* は、サンプリング周波数の分数で表されるフィルター設計周波数で、0.5（正規化ナイキスト周波数）を超えてはなりません。

許可されるフィルター長の最大値は 400,001 ポイントであり、これには *fw* >= 0.000789（サンプリング周波数の 0.0789%）が必要です。

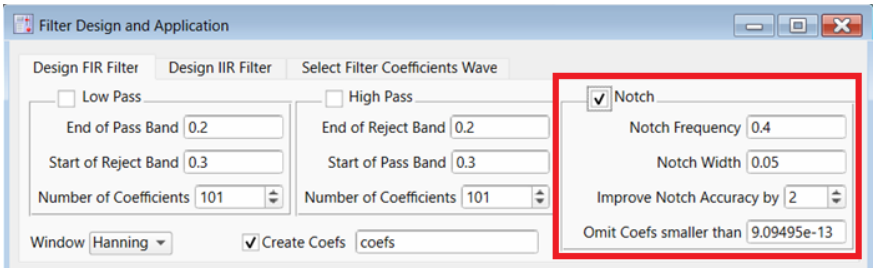
Igor Pro 8.03 以前のバージョンでは、フィルターの長さの最大値は 4,001 ポイントで、*fw* は 0.00789 以上（サンプリング周波数の 0.789%）でした。

オプションの *eps* パラメーターよりも小さい端部の係数は削除されるため、フィルターは短くなります（処理も高速化される）が、精度は低下します。デフォルト値は 2^-40 です。0 を指定すると、どんなに小さくても、さらにはゼロの係数であっても、すべて

の係数が保持されます。すべての係数を保持すると、*fw* が小さくなるにつれて実行時間が大幅に増加します。

nMult は、最も正確なノッチ周波数を得るために、フィルターの長さをどれだけ長くできるかを指定します。デフォルトは 2 です（係数の数が 2 倍になる可能性があります）。可能な限り短いフィルターを生成するには、*nMult* を 1 以下に設定してください。

最大平坦ノッチフィルターの設計は、Zahradník と Vlcek の研究に基づいており、係数の計算には任意精度演算（APMath コマンドを参照）が使われています。



/WINF=*windowKindName*

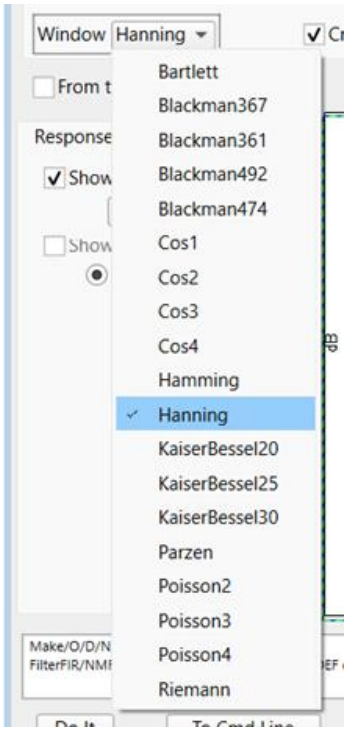
指定された「ウィンドウ」をフィルター係数に適用します。ウィンドウは、フィルターの周波数特性を顕著な形や微妙な形で変化させ、阻止帯域の遮断性能を高めたり、通過周波数と遮断周波数の間の遷移領域の傾斜を急峻にしたりします。フィルター係数が多数使われる場合は、ウィンドウの影響はそれほど重要ではありません。

/WINF が指定されていない場合は、Hamming ウィンドウが使われます。係数フィルタリングを行わない場合は、/WINF=None を指定してください。

windowKind の選択肢は以下の通りです：

Bartlett	Blackman367
Blackman361	Blackman492
Blackman474	Cos1
Cos2	Cos3
Cos4	Hamming
Hanning	KaiserBessel20
KaiserBessel25	KaiserBessel30
Parzen	Poisson2
Poisson3	Poisson4
Riemann.	

ウィンドウ方程式や詳細については、FFT コマンドを参照してください。



詳細

coefsWaveName が指定されている場合、/HI、/LO、/NMF は無視されます。

/HI、/LO、/NMF のうち 2 つ以上が指定された場合、フィルターは線形畳み込みを使って結合されます。結合されたフィルターの長さは、個々のフィルターの長さの合計よりわずかに短くなります。

/LO と /HI の両方が指定されると、バンドパスフィルターまたはバンドリジェクトフィルターが生成されます。
/LO 周波数が /HI 周波数よりも大きい場合、バンドパスフィルターが生成されます（ローパスフィルターとハイパスフィルターの通過帯域が重なります）。

Igor Pro 9.0 以降では、/LO 周波数が /HI 周波数よりも小さい場合、バンドリジェクトフィルターが生成されます（フィルタの阻止帯域が重なります）。

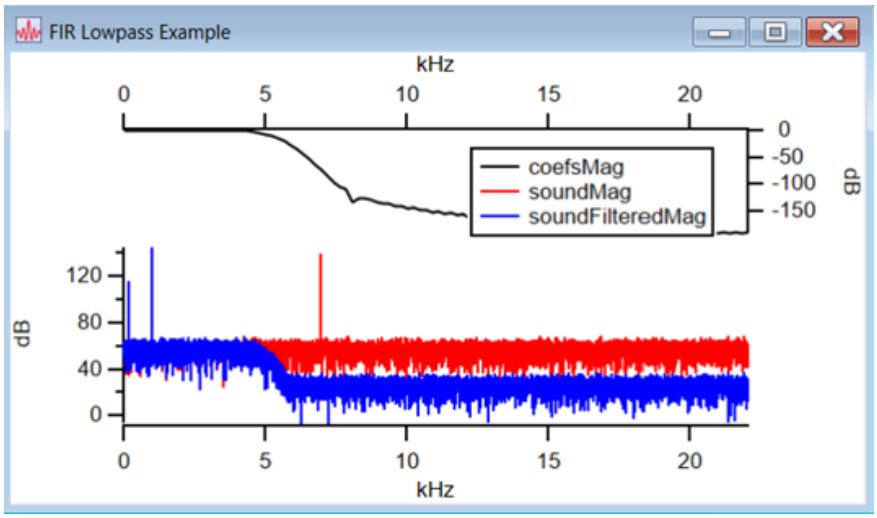
フィルタリング畳み込みは時間領域で行われます。
つまり、データのフィルタリングに FFT は使われません。
このため、係数の長さは対象となるウェーブに比べて短くする必要があります。

FilterFIR は、*coefsWaveName* の中間ポイントが delay=0 のポイントに対応すると仮定します。
「中間」ポイントの番号は、trunc(numpnts(coefsWaveName - 1)/2) となります。
coefsWaveName には通常、フィルターの双方向インパルス応答が含まれており、そのポイント数は通常、奇数です。
これは、/HI、/LO、/NMF によって生成される係数データの一種です。

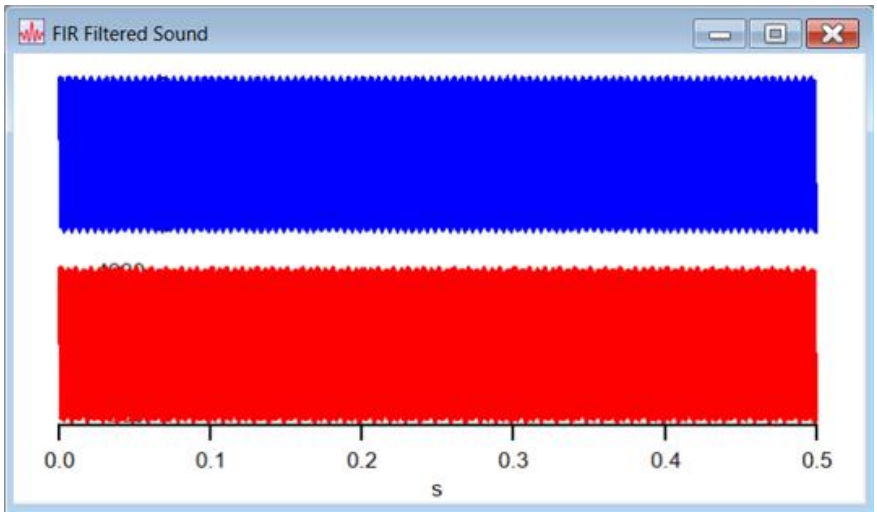
FilterFIR は、すべてのウェーブの X 軸スケーリングを無視します。
ただし、/COEF によって係数ウェーブが生成される場合は例外です。
この場合、X 軸スケーリングの deltax は保持され、leftx の値が変更されて、ゼロ位相（中心）の係数が x=0 に位置するように調整されます。

例

```
// 3つの正弦ウェーブからテスト音を作成する
Variable/G fs= 44100                        // サンプリング周波数
Variable/G seconds= 0.5                     // 長さ
Variable/G n= 2*round(seconds*fs/2)
Make/O/W/N=(n) sound                        // 16ビット整数ウェーブ
SetScale/p x, 0, 1/fs, "s", sound
Variable/G f1= 200, f2= 1000, f3= 7000
Variable/G a1=100, a2=3000,a3=1500
sound= a1*sin(2*pi*f1*x)
sound += a2*sin(2*pi*f2*x)
sound += a3*sin(2*pi*f3*x)+gnoise(10)       // ノイズフロアを追加

// その音のスペクトルをdB単位で算出
FFT/MAG/WINF=Hanning/DEST=soundMag sound
soundMag= 20*log(soundMag)
SetScale d, 0, 0, "dB", soundMag

// ウェーブに5kHzのローパスフィルターを適用
Duplicate/O sound, soundFiltered
FilterFIR/E=3/LO={4000/fs, 6000/fs, 101} soundFiltered

// フィルター処理後の音のスペクトルをdB単位で算出
FFT/MAG/WINF=Hanning/DEST=soundFilteredMag soundFiltered
soundFilteredMag= 20*log(soundFilteredMag)
SetScale d, 0, 0, "dB", soundFilteredMag

// フィルターの周波数特性をdB単位で計算
Make/O/D/N=0 coefs                          // 倍精度が推奨される
SetScale/p x, 0, 1/fs, "s", coefs
FilterFIR/COEF/LO={4000/fs, 6000/fs, 101} coefs
FFT/MAG/WINF=Hanning/PAD={(2*numpnts(coefs))}/DEST=coefsMag coefs
coefsMag= 20*log(coefsMag)
SetScale d, 0, 0, "dB", coefsMag

// 周波数特性をグラフ化
Display/R/T coefsMag as "FIR Lowpass Example";DelayUpdate
AppendToGraph soundMag, soundFilteredMag;DelayUpdate
ModifyGraph axisEnab(left)={0,0.6}, axisEnab(right)={0.65,1}
ModifyGraph rgb(soundFilteredMag)=(0,0,65535), rgb(coefsMag)=(0,0,0)
Legend
```




```
// フィルター処理前とフィルター処理後の音の時間応答をグラフ化
Display/L=leftSound sound as "FIR Filtered Sound";DelayUpdate
AppendToGraph/L=leftFiltered soundFiltered;DelayUpdate
ModifyGraph axisEnab(leftSound)={0,0.45}, axisEnab(leftFiltered)={0.55,1}
ModifyGraph rgb(soundFiltered)=(0,0,65535)

// 音の再生
PlaySound sound                 // これは高音
PlaySound soundFiltered         // これは少し低い音
```



グラフが表示されるとともに、２つの音が再生されます。

参考文献

Zahradník, P., and M. Vlcek, Fast Analytical Design Algorithms for FIR Notch Filters, IEEE Trans. on Circuits and Systems, 51, 608 - 623, 2004.

参照

Smooth、Convolve、Smoothing、MatrixConvolve、MatrixFilter コマンド
ヘルプ FIR Filters