

ΔΣツールボックス Version 7.1

本ΔΣツールボックスは、シグナルプロセッシングとコントロールシステムの2つのツールボックスを利用している。また、特定の関数 (clans や designLCBP) を使う場合は、オプティマイゼーション・ツールボックスも必要になる。本ΔΣツールボックスは、MathWorks のホームページ (<http://www.mathworks.com/matlabcentral/fileexchange>) の Control カテゴリから delsig を選択してダウンロードできる。シミュレーションの実行速度を上げたい場合は、simulationDSM.c をコンパイルしてほしい。MATLAB プロンプトのコマンドラインから mex simulationDSM.c と打ち込むことでコンパイルが行える。他の関数 simulationRSL.c や ai2mif.c も同様にコンパイルすることができる。具体的な関数の使用方法や実行例に関しては、以下の著作の第8章と9章を参考にしてほしい。

“Understanding Delta-Sigma Data Converters,” R.Schreier and G.C.Temes, John Wiley & Sons, New York, 2004. 訳書「ΔΣ型アナログ/デジタル変換器入門」和保 孝夫 / 安田 彰 (監訳) 丸善 2007.

ツールボックス内での注意事項.

周波数は正規化されている。すなわち $f = 1$ は f_s に対応している。

引数のデフォルト値は関数のパラメータリスト中の等号記号 (=) に続く値である。関数呼び出しの際にデフォルト値を使いたい場合は、引数がパラメータの終わりの場合には単純に引数を渡さなければ良い、それ以外の場合はNaN (数値以外) や [] (空行列) を引数として用いる。

なお、一般的な単一量子化器を持つΔΣ変調器のループフィルタを1つの行列に対応させている。詳しくは20ページの「変調器モデルの詳細」の中のABCD行列の説明を参考してほしい。

ツールボックスの実行例とΔΣ変調器の設計例に関して.

- dsdemo1 synthesizenTF 関数の実行例。5次のローパス変調器の雑音伝達関数の合成における零点の最適化の有り無しの実行例と、8次のバンドパス変調器の零点を最適化して合成した場合の実行例。
- dsdemo2 synthesizesDSM, predictSNR と simulationSNR 関数の実行例。時間領域のシミュレーションの実行例である。SNRの予想は「Ardalan and Paulos」法を用いており、スペクトラム解析とSNRの解析の様子を知ることができる。ローパス、バンドパスと多ビットローパス変調器の実行例が用意されている。
- dsdemo3 realizeNTF, stuffABCD, scaleABCD と mapABCD 関数の実行例。係数配列とスケール値の決定の実行例。
- dsdemo4 多段デシメーション sinc フィルタと変調器 MOD1, MOD2 の音声によるデモンストレーション。
- dsdemo5 simulateESL 関数の実行例。ミスマッチシェーピング DAC のエレメント選択の様子がわかる。
- dsdemo6 designHBF 関数の実行例。ハーフバンドフィルタのハードウェアを最適化する様子がわかる。
- dsdemo7 findPIS 関数の実行例。不変集合 (positively-invariant set) の計算例。
- dsdemo8 designLCBP 関数の実行例。連続時間のバンドパス変調器の設計例 (この関数の実行にはオプティマイゼーション・ツールボックスが必要である。)
- dsexample1 離散時間のローパス変調器の設計例。
- dsexample2 離散時間のバンドパス変調器の設計例。

主な関数

- ntf = synthesizenTF (order=3, R=64, opt=0, H_inf=1.5, f0=0) page 4
- ntf = clans (order=4, R=64, Q=5, rmax=0.95, opt=0) page 5

雑音伝達関数の合成.

[snr, amp, k0, k1, sigma_e2] = predictSNR(ntf, R=64, amp=..., f0=0) page 6

Ardalan and Paulos法を使った入力パワーに対するSNRの予測計算.

[v, xn, xmax, y] = simulateDSM(u, ABCD, nlev=2, x0=0) page 7

[v, xn, xmax, y] = simulateDSM(u, ntf, nlev=2, x0=0)

与えられた入力に対してΔΣ変調器のシミュレーションを実行.

[snr, amp] = simulateSNR(ntf, R, amp=..., f0=0, nlev=2, f=1/(4*R), k=13) page 8

入力パワーに対するSNRをシミュレーションにより計算する.

[a, g, b, c] = realizeNTF(ntf, form='CRFB', stf=1) page 9

雑音伝達関数を具体的な変調器の構成の係数に変換する.

ABCD = stuffABCD(a, g, b, c, form='CRFB') page 10

具体的な変調器の構成に対応したABCD行列を計算する.

[a, g, b, c] = mapABCD(ABCD, form='CRFB') page 10

ABCD行列で与えられた具体的な変調器の構成のパラメータを計算する

[ABCDs, umax] = scaleABCD(ABCD, nlev=2, f=0, xlim=1, ymax=nlev+2) page 11

ABCD行列で与えられたΔΣ変調器のダイナミックレンジ・スケーリングを行う.

[ntf, stf] = calculateTF(ABCD, k=1) page 12

ABCD行列で与えられたΔΣ変調器のNTFとSTFを量子化器のゲイン(k)を与えて計算する.

[sv, sx, sigma_se, max_sx, max_sy] =

simulateESL(v, mtf, M=16, dw=[1...], sx0=[0...]) page 13

ミスマッチシェーピングDACの素子選択回路をシミュレーションする.

[f1, f2, info] = designHBF(fp=0.2, delta=1e-5, debug=0) page 14

デシメーションもしくは補間フィルターを使ったハーフバンドフィルターのハードウェア最適化設計の実行.

[param, H, L0, ABCD, x] = page 17

designLCBP(n=3, OSR=64, opt=2, Hinf=1.6, f0=1/4, t=[0 1], form='FB', x0, dbg)

連続時間LCバンドパス変調器を設計する.

[s, e, n, o, Sc] = findPIS(u, ABCD, nlev=2, options) page 19

ΔΣ変調器の凸包不変集合を探す.

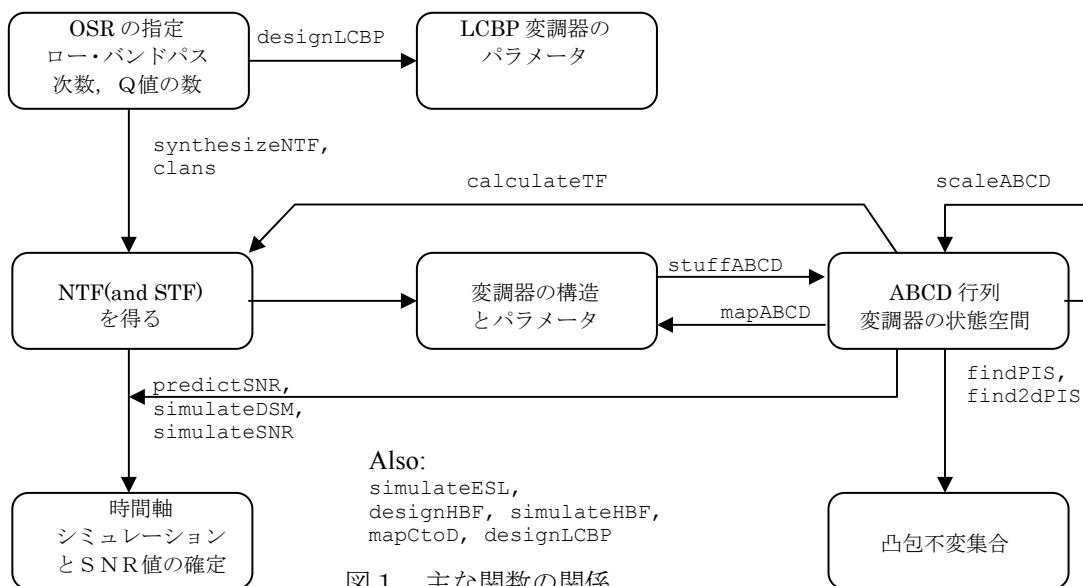


図 1 主な関数の関係

その他の関数

ΔΣ 関連のユーティリティ

mod1, mod2

一次と二次の変調器のNTFとSTFの設定用スクリプト.

snr = calculateSNR(hwfft, f)

与えられたハニング窓と入力信号の周波数を指定してSNR値を評価する.

[sys, Gp] = mapCtoD(sys_c, t=[0 1], f0=0)

連続時間系からインパルス応答が同じ不連続時間系への変換.

[A B C D] = partitionABCD(ABCD, m)

ABCD行列のパーティショニング.

H_inf = infnorm(H)

z軸の伝達関数の最大絶対値を計算する. evalTFを参照.

sigma_H = rmsGain(H, f1, f2)

離散時間の伝達関数Hの帯域内(f1, f2)でのRMSゲインの計算.

一般的なユーティリティ

dbv(), dbp(), undbv(), undbp(), dbm()

電圧, パワーのデシベル表記への変換と逆変換.

window = hann(N)

長さNのハニング窓関数. MATLABのhanning 関数とは異なりFFTの繰り返しサイクル端に影響を及ぼさない(繰り返し周波数の整数倍の波長に影響はない). MATLAB6の関数

hann(N, 'predict') と hann(N) は同じ.

表示関係のユーティリティ

plotPZ(H, color='b', markersize=5, list=0)

伝達関数の極と零点を表示する.

figureMagic(xRange, dx, xLab, yRange, dy, yLab, size)

グラフの軸, 上下限やラベル等, グラフ表示用の簡易コマンド.

printmif(file, size, font, fig)

グラフをAdobeのイラストレーターのフォーマットに変換する. 変換後はai2mif関数により

Adobeのフレームメーカーのフォーマットに変換できる. ai2mif関数はDeron Jackson

djackson@mit.eduが作った同じ名前の関数の改良版である.

[f, p] = logsmooth(X, inBin, nbin)

FFTの結果を平滑化してdB値に変換する. bplotsmoothとbilogplotを参考.

synthesizeNTF

使い方: `ntf = synthesizeNTF(order=3,OSR=64,opt=0,H_inf=1.5,f0=0)`
 ΔΣ変調器の雑音伝達関数を(NTF) 計算する.

引数

- order* NTFの次数. *order* バンドパス変調器の場合は必ず偶数.
- OSR* オーバーサンプリング比. *OSR* はNTFの零点を最適化するには必須.
- opt* NTFの極を最適化する際のフラグ. *opt=0* ではすべてのNTFの零点をバンド中心 (ローパスの場合DC) に配置する. *opt=1* の場合はNTFの零点の最適化を行う. 奇数の次数の変調器の場合, *opt=2* を指定すると2つの零点をバンド中心に配置し, それ以外のものを最適化する.
- H_inf* NTFの帯域外利得. 「リーの基準」から, バイナリ量子化器の場合 $H_inf < 2$ であれば変調器は安定である. *H_inf* を小さくすることで多くの場合うまく行くが, NTFでの減衰が大きくなるので変調器の理論的な分解能は減少する.
- f0* 変調器の中心周波数. $f0 \neq 0$ はバンドパス変調器になる. $f0=0.25$ では中心周波数を $f_s/4$ にしたことになる.

出力

ntf 線形時不変の零点-極-ゲイン形式のNTF値

Example

5次のローパス変調器. OSRを32にして零点を最適化する.

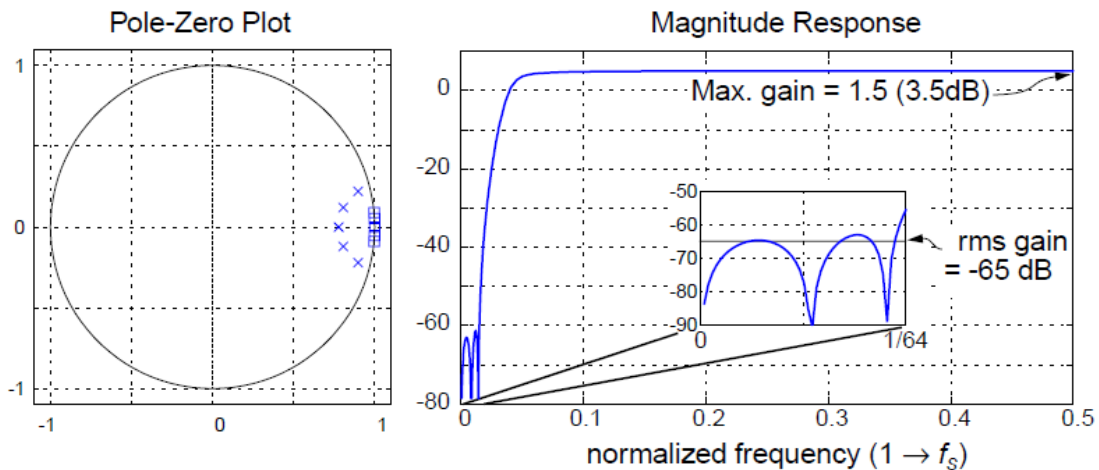
>> `synthesizeNTF(5,32,1)`

零点/極/ゲイン:

$(z-1) (z^2 - 1.997z + 1) (z^2 - 1.992z + 1)$

 $(z-0.7778) (z^2 - 1.613z + 0.6649) (z^2 - 1.796z + 0.8549)$

サンプル時間: 1



clans

使い方: `ntf = clans(order=4,OSR=64,Q=5,rmax=0.95,opt=0)`

CLANS(Closed-loop analysis of noise-shaper)法[1]を使ってローパスΔΣ変調器の雑音伝達関数を求める。この関数の実行には最適化・ツールボックスが必要である。

[1] J. G. Kenney and L. R. Carley, "Design of multibit noise-shaping data converters," *Analog Integrated Circuits Signal Processing Journal*, vol. 3, pp. 259-272, 1993.

引数

- order* NTFの次数.
- OSR* オーバーサンプリング比.
- Q* 量子化雑音をフィードバックする量子化器のレベル数(数学的には $Q = \|h\|_1 - 1$, つまりインパルス応答の絶対値を加算した値から1を引いたもの, すなわち最大瞬間雑音ゲインである).
- rmax* NTFの極の最大半径.
- opt* NTFの極を最適化する際のフラグ. *opt=0* ではすべてのNTFの零点をバンド中心(ローパスの場合DC)に配置する. *opt=1* の場合はNTFの零点の最適化を行う. 奇数の次数の変調器の場合, *opt=2* を指定すると2つの零点をバンド中心に配置し, それ以外のものを最適化する.

出力

ntf 線形時不変の零点-極-ゲイン形式のNTF値

Example

5次のローパス変調器. 雑音ゲインが5,OSRを32にして零点を最適化する.

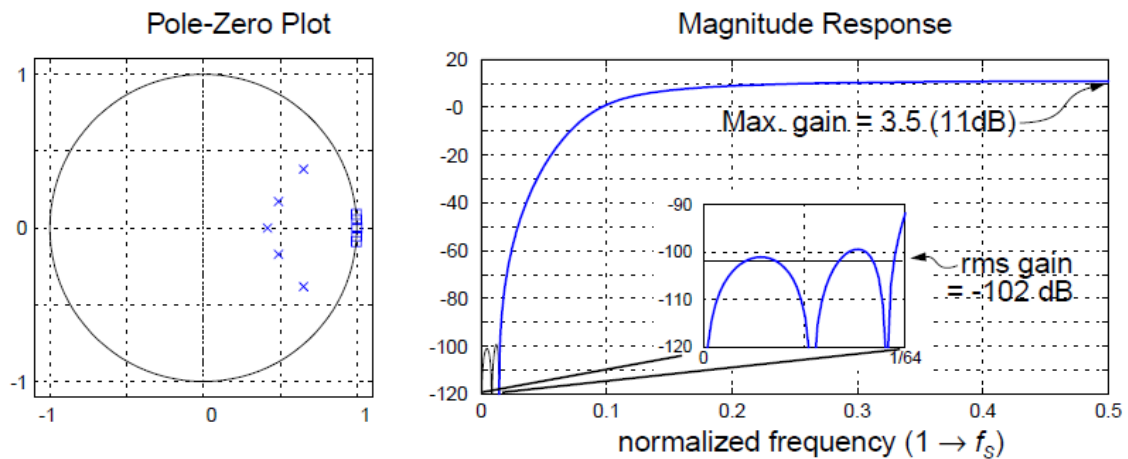
```
>> H= clans(5,32,5,.95,1)
```

零点/極/ゲイン:

$$(z-1) (z^2 - 1.997z + 1) (z^2 - 1.992z + 1)$$

$$(z-0.4184) (z^2 - 0.9784z + 0.2686) (z^2 - 1.305z + 0.5714)$$

サンプリング時間: 1



predictSNR

使い方: `[snr, amp, k0, k1, sigma_e2] = predictSNR(ntf, OSR=64, amp=..., f0=0)`

「Ardalan and Paulos [1]」法を使って異なる入力振幅に対するSNR値を予測する。但し、この予測手法はバイナリ変調器にのみ有効である。 [1] S. H. Ardalan and J. J. Paulos, "Analysis of nonlinear behavior in delta-sigma modulators," *IEEE Transactions on Circuits and Systems*, vol. 34, pp. 593-603, June 1987.

引数

- ntf* 零点-極-ゲイン形式の変調器のNTF値.
- OSR* オーバーサンプリング比. *OSR* は注目帯域を定めるために使われる.
- amp* 必要な増幅度の列ベクトル記述. デフォルト値は[-120 -110...-20 -15 -10 -9 -8 ... 0] dB, ここで 0 dB はフルスケール (ピーク=1) のサイン波の意味.
- f0* 変調器の中心周波数. *f0* ≠ 0 はバンドパス変調器になる.

出力

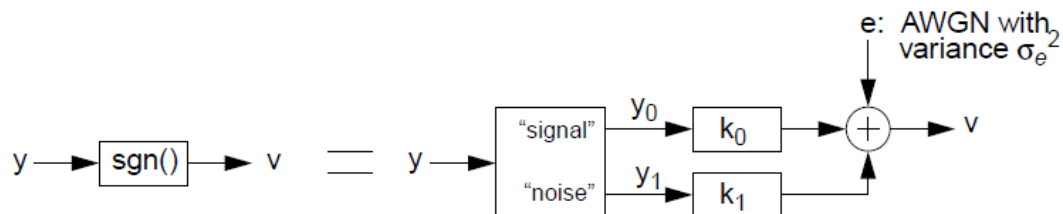
- snr* SNRの予測値が格納された列ベクトル.
- amp* 増幅度の列ベクトル.
- k0* 入力振幅に対応した量子化器モデルの信号増幅率.
- k1* 入力振幅に対応した量子化器モデルの雑音増幅率.
- sigma_e2* 量子化器モデルからのノイズの二乗平均平方根(RMS).

Example

8 ページの例を参照.

量子化器モデル:

以下の図にあるように、量子化器モデルは線形増幅器と雑音源の対で構成されている。量子化器に入力された信号はそれぞれ信号振幅に応じた増幅率 k_0 と k_1 を掛け合わされて信号成分と雑音成分に分けられる。量子化器の出力ではさらに信号振幅に応じた分散値を持つ白色ガウス雑音を加えられている。



simulateDSM

使い方: `[v,xn,xmax,y] = simulateDSM(u,ABCD,nlev=2,x0=0)` or

`[v,xn,xmax,y] = simulateDSM(u,ntf,nlev=2,x0=0)`

与えられた入力を使いΔΣ変調器をシミュレートする。コンパイルされた高速版を使う際はmexファイルがサーチパスにあるかを確認する(MATLABプロンプトでwhich simulateDSMとタイプする)。

引数

u 列ベクトル記述の変調器に入力される $m \times N$ 信号列。*m* は入力信号の数(普通は1)でフルスケールは*nlev-1*に対応する。

ABCD 変調器のループフィルターのABCD行列(状態空間記述)。

ntf 零点-極-ゲイン形式の変調器のNTF値。変調器のSTFは1を仮定。

nlev 量子化器のレベル数。複数の量子化器は*nlev*を行列で記述する。

x0 変調器の初期状態。

出力

v 入力信号に対応した変調器の出力。

xn 入力信号に対応した行列形式の変調器の内部状態。

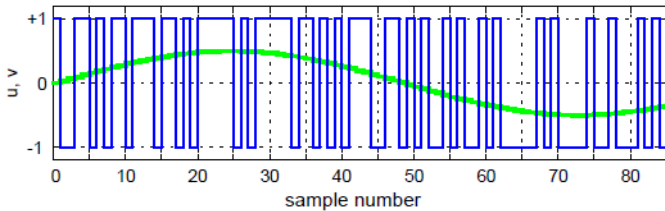
xmax 状態変数の最大値。

y 入力信号に対応した量子化器への入力値。

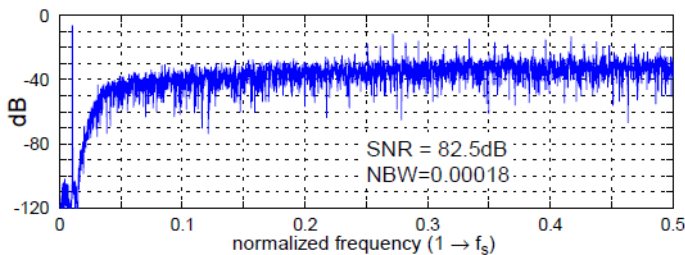
Example

5次のローパス変調器に1/2の振幅のサイン波を入力した場合の出力の時間軸と周波数軸のシミュレーション結果。

```
>> OSR = 32; H = synthesizENTF(5,OSR,1);
>> N = 8192; fB = ceil(N/(2*OSR)); f=85; u = 0.5*sin(2*pi*f/N*[0:N-1]);
>> v = simulateDSM(u,H);
```



```
t = 0:85;
stairs(t, u(t+1));
hold on;
stairs(t, v(t+1));
axis([0 85 -1.2 1.2]);
ylabel('u, v');
```



```
spec=fft(v.*hann(N))/(N/4);
plot(linspace(0,1,N/2),
dbv(spec(1:N/2)));
axis([0 1 -120 0]);
grid on;
ylabel('dB')
snr=calculateSNR(spec(1:fB),f);
s=sprintf('SNR = %4.1fdB\n',snr)
text(0.5,-90,s);
s=sprintf('NBW=%7.5f',1.5/N);
text(0.5,-110,s);
```

simulateSNR

使い方: `[snr,amp] = simulateSNR(ntf,OSR,amp,f0=0,nlev=2,f=1/(4*OSR),k=13)`
 異なるサイン波の入力振幅を入力としてΔΣ変調器をシミュレートしてそれぞれの入力振幅に対するSNR値を計算する。

引数

- ntf* 零点-極-ゲイン形式の変調器のNTF値.
- OSR* オーバーサンプリング比. *OSR* は注目帯域を定めるために使われる.
- amp* 必要な増幅度の列ベクトル記述. デフォルト値は[-120 -110...-20 -15 -10 -9 -8 ... 0] dB, ここで 0 dB はフルスケール (ピーク値=*nlev*-1) のサイン波の意味.
- f0* 変調器の中心周波数.*f0* ≠ 0 はバンドパス変調器になる.
- nlev* 量子化器のレベル数.
- f* 正規化された入力サイン波の周波数, 注目帯域中であること. 周波数はFFTの繰り返しに丁度収まるよう調整する.
- k* FFTのポイント数は 2^k になる

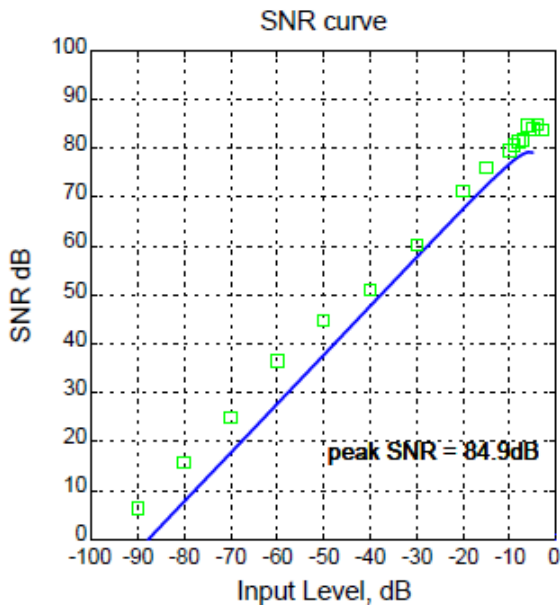
出力

- snr* シミュレートされたSNRの値が格納された列ベクトル.
- amp* 計算に使われた増幅度が格納された列ベクトル.

Example

5次の変調器の入力振幅とSNRの関係の予測値とシミュレーション値の比較.

```
>> OSR = 32; H = synthesizENTF(5,OSR,1);
>> [snr_pred,amp] = predictSNR(H,OSR);
>> [snr,amp] = simulateSNR(H,OSR);
```



```
plot(amp,snr_pred,'b',amp,snr,'gs');
grid on;
figureMagic([-100 0], 10, 1, ...
[0 100], 10, 1);
xlabel('Input Level, dB');
ylabel('SNR dB');
title('SNR curve');
s=sprintf('peak SNR = %4.1fdB\n',...
max(snr));
text(-49,15,s);
```

realizeNTF

使い方: `[a,g,b,c] = realizeNTF(ntf,form='CRFB',stf=1)`

雑音伝達関数(NTF)を特定の変調器の構成の係数に変換する。

引数

ntf 零点-極-ゲイン形式の変調器のNTF値.
form 特定の変調器の構成に対応した文字列.
 CRFB Cascade-of-resonators, feedback form.
 CRFF Cascade-of-resonators, feedforward form.
 CIFB Cascade-of-integrators, feedback form.
 CIFF Cascade-of-integrators, feedforward form.

変調器の構成の詳細に関しては20ページを参照。

stf 零点-極-ゲイン形式の変調器のSTF。新たな状態変数を導入せず与えられたSTFを実現するためにSTFの極はNTFの極と一致していることに注意。

出力

a 量子化からのフィードバック/フィードフォワードの係数 ($1 \times n$).
g 共振器の係数 ($1 \times \lfloor n/2 \rfloor$).
b 変調器から積分器への入力係数 ($1 \times n + 1$).
c 積分器間の係数 (スケールされていない変調器では*c*はすべて1) ($1 \times n$)

Example

CRFB構造の5次の変調器の係数を求める。

```
>> H = synthesizеNTF(5,32,1);
>> [a,g,b,c] = realizeNTF(H,'CRFB')
a =
    0.0007    0.0084    0.0550    0.2443    0.5579
g =
    0.0028    0.0079
b =
    0.0007    0.0084    0.0550    0.2443    0.5579    1.0000
c =
     1     1     1     1     1
```

stuffABCD

使い方 `ABCD = stuffABCD(a, g, b, c, form='CRFB')`

ABCD行列を特定の変調器の構成の係数から計算する。

引数

- a* 量子化からのフィードバック/フィードフォワードの係数 ($1 \times n$).
- g* 共鳴器の係数 ($1 \times \lfloor n/2 \rfloor$).
- b* 変調器から積分器への入力係数 ($1 \times n + 1$).
- c* 積分器間の係数 (スケールされていない変調器では*c*はすべて1) ($1 \times n$).
- form* 9ページのrealizeNTFの説明中の変調器の構成を参照.

出力

ABCD 変調器のループフィルターのABCD行列.

mapABCD

使い方 `[a, g, b, c] = mapABCD(ABCD, form='CRFB')`

特定の変調器の構成の係数を特定の変調器の構成のABCD行列から計算する.

引数

- ABCD* 変調器のループフィルターのABCD行列.
- form* 9ページのrealizeNTFの説明中の変調器の構成を参照.

出力

- a* 量子化からのフィードバック/フィードフォワードの係数 ($1 \times n$).
- g* 共鳴器の係数 ($1 \times \lfloor n/2 \rfloor$).
- b* 変調器から積分器への入力係数 ($1 \times n + 1$).
- c* 積分器間の係数 (スケールされていない変調器では*c*はすべて1) ($1 \times n$).

scaleABCD

使い方: `[ABCDs, umax]=scaleABCD(ABCD, nlev=2, f=0, xlim=1, ymax=nlev+5, umax, N=1e5)`

ABCD行列の状態変数の最大値が与えられた制限値を超えないようにスケーリングする。安定な入力振幅の最大値も副次的に計算される。

引数

ABCD 変調器のループフィルターのABCD行列。
nlev 量子化器のレベル数。
f テスト用サイン波の正規化周波数。
xlim 状態変数の制限値（殆どの場合配列値で与えられる）。
ymax 変調器の安定度を定める閾値。量子化器の入力が*ymax*を越えると変調器は不安定だと判断する。

出力

ABCDs スケールされた変調器のループフィルターのABCD行列。
umax 安定な入力振幅の最大値。この値以下のサイン波が入力さえしていれば、変調器が与えられた制限値を超えることはない。

calculateTF

使い方: [ntf,stf] = calculateTF(ABCD,k=1)

ΔΣ変調器のNTFとSTFを計算する.

引数

ABCD 変調器のループフィルターのABCD行列.

k 量子化器のゲイン.

出力

ntf 線形時不変の零点-極-ゲイン形式のNTF値.

stf 線形時不変の零点-極-ゲイン形式のSTF値.

Example

CRFB構造の5次の変調器を設計して、ループフィルターのABCD行列とNTFとSTFの値を検証する.

```
>> H = synthesizENTF(5,32,1)
```

零点/極/ゲイン:

$$(z-1) (z^2 - 1.997z + 1) (z^2 - 1.992z + 1)$$

$$(z-0.7778) (z^2 - 1.613z + 0.6649) (z^2 - 1.796z + 0.8549)$$

サンプリング時間: 1

```
>> [a,g,b,c] = realizeNTF(H)
```

a =

```
0.0007 0.0084 0.0550 0.2443 0.5579
```

g =

```
0.0028 0.0079
```

b =

```
0.0007 0.0084 0.0550 0.2443 0.5579 1.0000
```

c =

```
1 1 1 1 1
```

```
>> ABCD = stuffABCD(a,g,b,c)
```

ABCD =

```
1.0000 0 0 0 0 0.0007 -0.0007
1.0000 1.0000 -0.0028 0 0 0.0084 -0.0084
1.0000 1.0000 0.9972 0 0 0.0633 -0.0633
0 0 1.0000 1.0000 -0.0079 0.2443 -0.2443
0 0 1.0000 1.0000 0.9921 0.8023 -0.8023
0 0 0 0 1.0000 1.0000 0
```

```
>> [ntf,stf] = calculateTF(ABCD)
```

零点/極/ゲイン:

$$(z-1) (z^2 - 1.997z + 1) (z^2 - 1.992z + 1)$$

$$(z-0.7778) (z^2 - 1.613z + 0.6649) (z^2 - 1.796z + 0.8549)$$

サンプリング時間: 1

零点/極/ゲイン:

1

simulateESL

使い方: [sv, sx, sigma_se, max_sx, max_sy]

= simulateESL(v, mtf, M=16, dw=[1...], sx0=[0...])

ミスマッチシェーピング伝達関数(mtf)を使い、複数素子DACの選択ロジックをシミュレーションする。

[1] R. Schreier and B. Zhang “Noise-shaped multibit D/A convertor employing unit elements,” *Electronics Letters*, vol. 31, no. 20, pp. 1712-1713, Sept. 28 1995.

引数

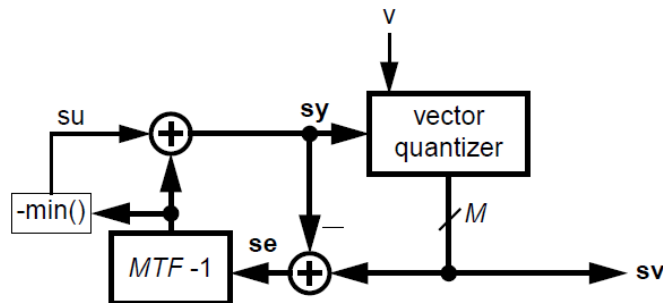
- v 同時に活性化する素子の数. simulateDSMからの出力は, $[0, \sum_i^M dw(i)]$ の範囲内に必ず入るよう調整 (オフセット・スケーリング) さえていなければならない.
- mtf 零点-極-ゲイン形式のMTF値.
- M DACの素子数.
- dw それぞれの素子に対応した重み配列.
- sx0 DACの選択ロジックの初期値の $n \times M$ 行列.

出力

- sv 選択配列, 0 もしくは 1 は素子の活性化に対応している.
- sx DACの選択ロジックの最終状態の配列.
- sigma_se 選択エラーのrms値 $se = sv - sy$. *sigma_se* は解析的に素子のミスマッチによる帯域内のノイズを予測する際に使われる.
- max_sx 素子選択ロジックが達した最大値.
- max_sy ベクトル量子化器への入力の最大値.

Example

dsdemo5.m.を実行.



Block diagram of the Element Selection Logic

designHBF

使い方: [f1,f2,info]=designHBF(fp=0.2,delta=1e-5,debug=0)

ΔΣ変調器とともに使われるデシメーションもしくは補間型の線形位相ハーフバンドフィルターの最適化ハードウェアの設計。この関数はSaramäki [1]に記載されている方法に基づいている。この手法は非確定的な探索アルゴリズムに基づいているので、設計結果は試行毎に異なる可能性がある。[1] T. Saramäki, "Design of FIR filters as a tapped cascaded interconnection of identical subfilters," *IEEE Transactions on Circuits and Systems*, vol. 34, pp. 1011-1029, 1987.

引数

fp 正規化した通過帯域周波数。
delta 通過帯域と阻止帯域のリップルの絶対値。

出力

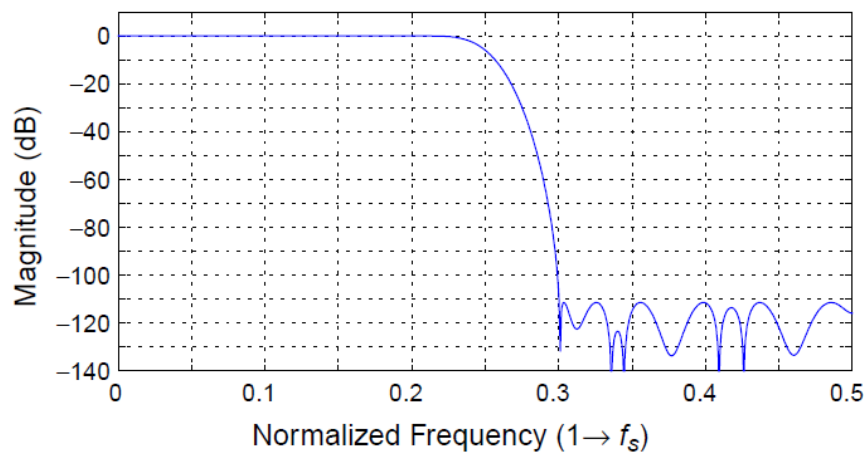
f1,f2 フィルターとサブフィルターの係数の最小重み係数(csd)表記。
info 以下の情報を含んだ配列(但しdebug=1の時だけ):
complexity 必要な加算器の数。
n1,n2 f1とf2配列の長さ。
sbr 阻止帯域の減衰(dB)。
phi f2フィルターのスケール値。

Example

通過帯域のリップルは 10^{-5} 以下で、阻止帯域の減衰量が少なくとも 10^{-5} (-100 dB)以上の、カットオフ周波数が0.2fsのローパスハーフバンドフィルターを設計する、

```
>> [f1,f2] = designHBF(0.2,1e-5);
>> f = linspace(0,0.5,1024);
>> plot(f, dbv(frespHBF(f,f1,f2)))
```

フィルターの周波数応答のプロットを以下に示す。124個の加算器（乗算器はなし）で阻止帯域の減衰量が109dBのフィルター特性が得られた。



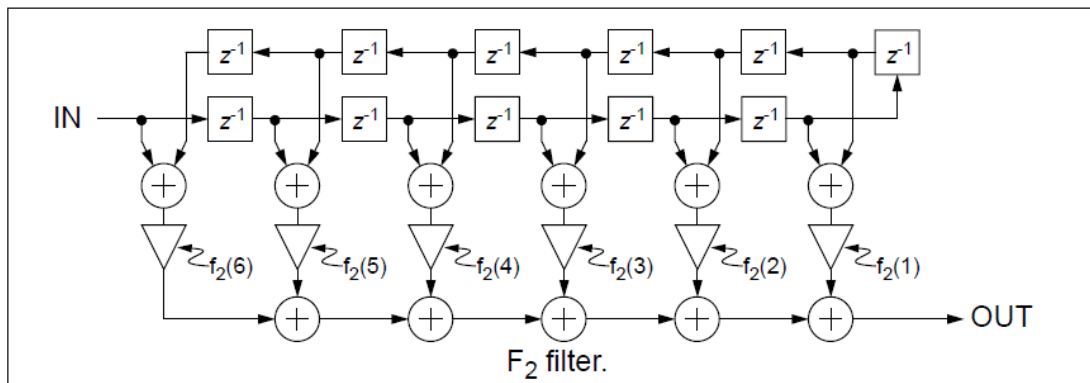
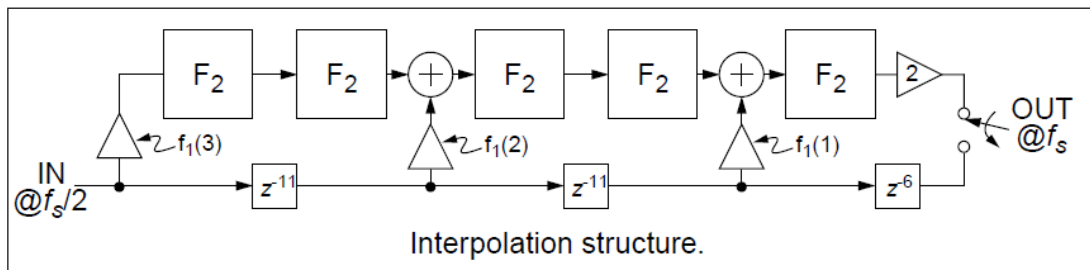
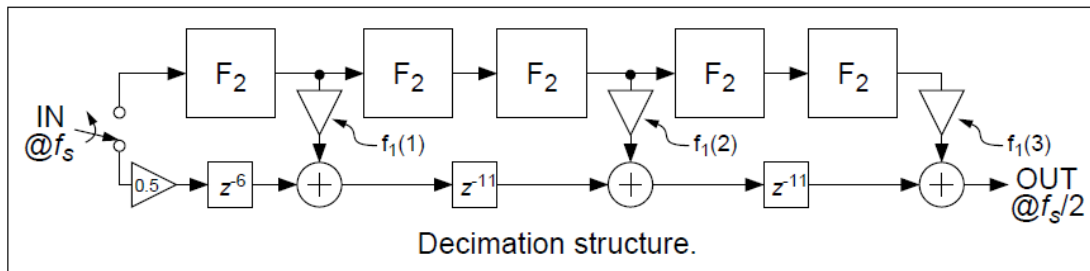
このフィルターのデシメーションもしくは補間形式を以下に示す。フィルターのSignedDigit表記。

```

[f1.val]' =      [f2.val]' =      >> f1.csd          >> f2.csd
  0.9453          0.6211          ans =           ans =
 -0.6406         -0.1895         0   -4   -7       -1  -3  -8
  0.1953          0.0957         1   -1   1       1   1  -1
                                ans =           -2  -4  -9
                                -1  -3   -6       -1  1  -1
                                -1  -1   -1       ans =           -3  -5  -9
                                -2  -4   -7       1  -1  1
                                1   -1   1       ans =           -4  -7  -8
                                                -1  1  1
                                                ans =           -5  -8 -11
                                                1  -1 -1
                                                ans =           -6  -9 -11
                                                -1  1  -1
    
```

このSignedDigit表記では、第一列が2のべき数を表し、第2列が符号を示している。

例えば $f(1)=0.9453=2^0-2^{-4}+2^{-7}$ で $f(2)=0.6211=2^{-1}+2^{-3}-2^{-8}$ になる。以下のダイアグラムの乗加算演算のフィルターの係数は、本例では3つのSignedDigitで与えられているので、高々3個の加算器が必要だけである。結果、この110次のFIRフィルターに必要な加算器は高々 $3 \cdot 3 + 5 \cdot (3 \cdot 6 + 6 - 1) = 124$ 個になる。



simulateHBF

使い方: `y = simulateHBF(x, f1, f2, mode=0)`

Saramakiのハーフバンドフィルタ(14ページdesignHBF参照)の時間解析シミュレーションを行う。

引数

- `x` 入力データ列.
- `f1, f2` designHBF の戻り値と同形式の、フィルタf1とf2の係数の配列もしくは構造体.
- `mode` 入力をフィルタするのか、補間するのか、それとも間引きするのかのフラグ。以下参照:
 - 0 フィルタのみ。補間も間引きもしない。
 - 1 入力は補間される。
 - 2 偶数サンプルが間引きされる。
 - 3 奇数サンプルが間引きされる。

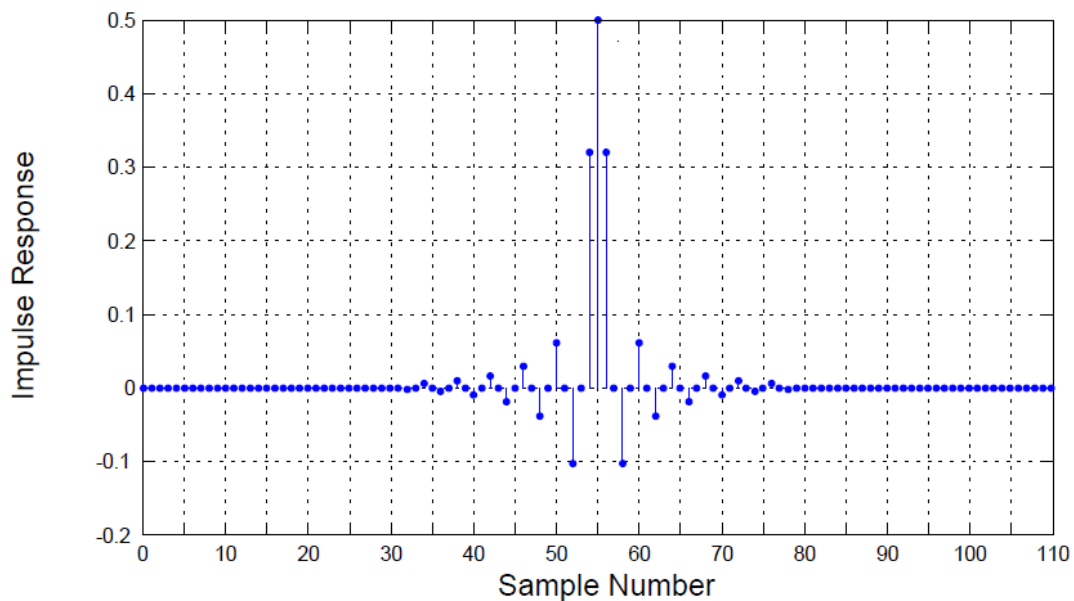
出力

- `y` 出力

Example

前ページで設計したフィルタの特性をシミュレーションする。

```
>> N = (2*length(f1)-1)*2*(2*length(f2)-1)+1;
>> y = simulateHBF([1 zeros(1,N-1)], f1, f2);
>> stem([0:N-1], y);
>> figureMagic([0 N-1], 5, 2, [-0.2 0.5], 0.1, 1)
>> printmif('HBFimp', [6 3], 'Helvetica8')
```



designLCBP

使い方: [param, H, L0, ABCD, x]=

designLCBP(n=3, OSR=64, opt=2, Hinf=1.6, f0=1/4, t=[0 1], form='FB', x0, dbg)

電流DACとトランスコンダクタで制御されたLCタンク回路と抵抗器からなる、連続時間のLCバンドパス変調器の設計をする。ダイナミックレンジとインピーダンスのスケールリングには対応していない。

引数

- n* ループフィルタ内のLCタンクの数。
- OSR* オーバーサンプリング比。 $OSR=f_s/(2f_B)$
- opt* NTFの零点を最適化する際のフラグ。
- H_inf* NTFの帯域外利得。4ページのsynthesizeNTFを参照。
- f0* 変調器の中心周波数。 default= $f_s/4$ 。
- t* DACのフィードバックパルスのスタートとエンド時間。 $t_1=0$ は遅延なし（つまり非現実的な仮定）で比較器を動作させることを意味する。
- form* 変調器の構成。以下のダイアグラム中の表を参照。
- dbg* 最適化の仮定を観察する場合は0以外の値を入れる。

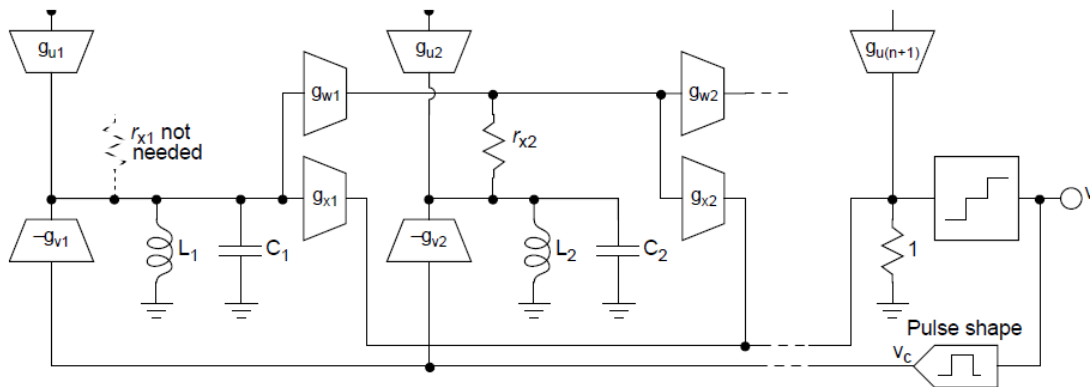
出力

- param* 以下のダイアグラムの表の引数とn, OSR, Hinf, f0, tとformの値が含まれる構造体。
- H* 変調器の離散時間の等価NTF。
- L0* 線形時不変の零点-極-ゲイン形式の連続時間変調器。

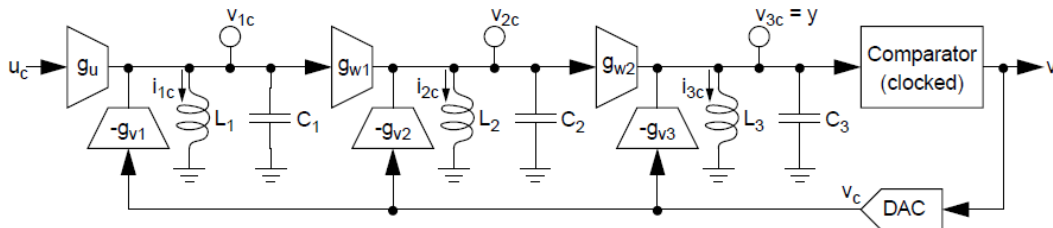
一般的なLC構造とそれぞれの構造に対応した最適化係数の一覧

Form	L	C	$g_u: 1 \times (n+1)$	$g_v: 1 \times n$	$g_w: 1 \times (n-1)$	$g_x: 1 \times n$	$r_x: 1 \times n$
FB	$\frac{1}{\sqrt{2\pi f_0}}$		[1 0 ...]*	[$x_1 x_2 \dots x_n$]	[1...]	[0 0... 1]	[0 ...]
FF			[1 0 ... 1]*	[1 0 ...]	[1...]	[$x_1 x_2 \dots x_n$]	[0 ...]
R			[1 0 ... 1]*	[$x_1 0 \dots$]	[1...]	[0 0... 1]	[0 $x_2 \dots x_n$]

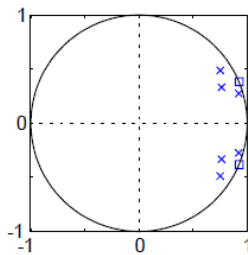
* scaled for unity STF gain at f_0 .



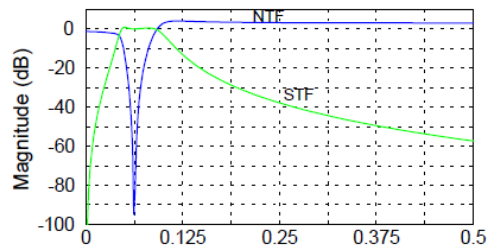
Example three-tank LC bandpass modulator (FB structure)



NTF Pole-Zero Plot



NTF and STF Frequency Response



Example

```
>> n = 3; OSR = 64; opt = 0; Hinf = 1.6; f0 = 1/16; t = [0.5 1]; form = 'FB';
>> [param,H,L0,ABCD,x] = designLCBP(n,OSR,opt,Hinf,f0,t,form);
>> plotPZ(H);
>> f = linspace(0,0.5,300); z = exp(2*pi*j*f);
>> ntf = dbv(evalTF(H,z)); stf = dbv(evalTFP(L0,H,f));
>> plot(f, ntf,'b', f, stf,'g');
>> figureMagic([0,0.5],1/16,2, [-100 10],10,2);
gu = 1 0 0 0
gv = 0.653 2.703 3.708
gw = 1 1
gx = 0 0 1
rx = 0 0 0
```

参考

```
[H,L0,ABCD,k]=LCparam2tf(param,k=1)
```

この関数はLCタンクを使ったΣ変調器でのABCD行列、量子化器のゲインと伝達関数を計算する。インダクターの直列抵抗(r_l) とキャパシターの短絡コンダクタンス(g_c)を考慮して計算することも出来る。入力と出力トランスコンダクタの有限のコンダクタンス値はこの g_c を接地して実現できる。

Bugs

designLCBPの最適化ではオブティマイゼーション・ツールボックス(version5と6)のconstr/fmincon関数を無理やり使っている。そのためLCobj*関数を編集することが時々必要になる。ステップサイズを制御出来るような頑強な最適化関数が必要である。designLCBPの構成は少々時代遅れになっている。より一般的なバックエンド構造のLC変調器が開発されている[1]ので文献を参考にしてほしい。

[1] R. Schreier, J. Lloyd, L. Singer, D. Paterson, M. Timko, M. Hensley, G. Patterson, K. Behel, J. Zhou and W. J. Martin, "A 10-300 MHz IF-digitizing IC with 90-105 dB dynamic range and 15-333 kHz bandwidth," *IEEE Journal of Solid-State Circuits*, vol. SC-37, no. 12, pp. 1636-1644, Dec. 2002.

findPIS, find2dPIS (PosInvSetディレクトリに格納)

```

使い方: [s,e,n,o,Sc] = findPIS(u,ABCD,nlev=2,options)
options = [dbg=0 itnLimit=2000 expFactor=0.005 N=1000 skip=100
           qhullArgA=0.999 qhullArgC=.001]
s = find2dPIS(u,ABCD,options)
options = [dbg=0 itnLimit=100 expFactor=0.01 N=1000 skip=100]

```

ΔΣ変調器の凸包不変集合を見つける。findPISはコンパイルしたqhull mexファイルが必要である。find2dPISにはその必要がないが、利用は2次のシステムに制限される。

引数

u 変調器の入力。 *u* がスケール値なら変調器の入力は一定である。もし *u* が 2×1 の配列値なら変調器の入力は 2×1 レンジ内でサンプルされた任意のシーケンスとなる。

ABCD 変調器のループフィルターのABCD行列。

nlev 量子化器のレベル数。

dbg *dbg*=1にすると繰り返し工程のグラフィック表示になる。

itnLimit 繰り返し回数の最大値

expFactor すべてのマッピング動作の前の拡張係数。 *expFactor* を増加すると繰り返し数は減るが、計算結果は期待したものより、たぶん大きくなる。

N 初期値を計算する際のポイント数。

skip 状態を観察する前に変調器を動作させる回数。これにより変調器の過渡確率を扱える。

qhullArgA *qhull*関数へ渡すA'引数。近傍の面の接線が、この値の絶対値よりも大きなコサイン角度を持つ場合、その面はマージされる。負の値はマージ動作がhullの構築の間で行われることを示し、正の値はポストプロセスで行われることを示している。

qhullArgC *qhull*関数へ渡すC'引数。近傍の面のcentroidとhyperplaneの間の距離が、この値の絶対値よりも小さい場合、その近傍の面はマージされる。負の値はマージ動作が前プロセスで行われることを示し、正の値は後プロセスで行われることを示している。

出力

s 頂点の集まり ($\text{dim} \times n_v$)。

e 頂点のペアでリストされたエッジの集まり ($2 \times n_e$)。

n 面の接線の集まり ($\text{dim} \times n_f$)。

o 面の垂線の集まり ($1 \times n_f$)。

Sc 内部で集合の丸めに使われるスケーリング行列。

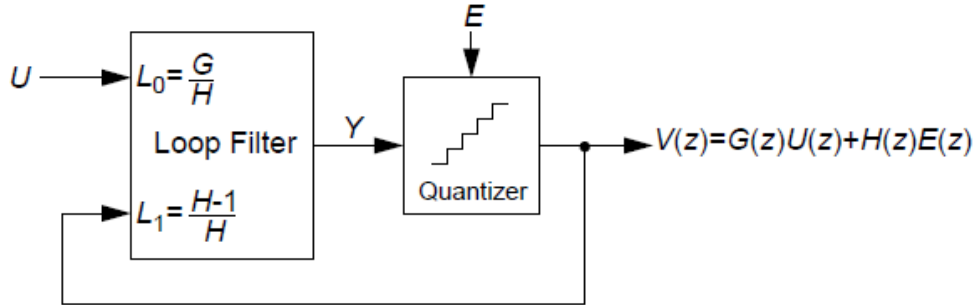
Background

以下の文献[1]で説明されている方法である。

- [1] R. Schreier, M. Goodson and B. Zhang "An algorithm for computing convex positively invariant sets for delta-sigma modulators," *IEEE Transactions on Circuits and Systems I*, vol. 44, no. 1, pp. 38-44, January 1997.

変調器モデルの詳細

以下のダイアグラムに示すように、単一量子化器を持つΔΣ変調器は、ループフィルタと量子化器で構成されているとする。



ループフィルタ

ループフィルタはABCD行列で記述する。単一量子化器からなるシステムの場合、ループフィルタは、2入力1出力の線形システムでありABCDは $(n+1) \cdot (n+2)$ 行列になる。以下に示すように $A(n \cdot n)$, $B(n \cdot 2)$, $C(1 \cdot n)$ and $D(1 \cdot 2)$ に分割して

$$ABCD = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \tag{1}$$

ループフィルタの状態と出力は以下のように計算できる。

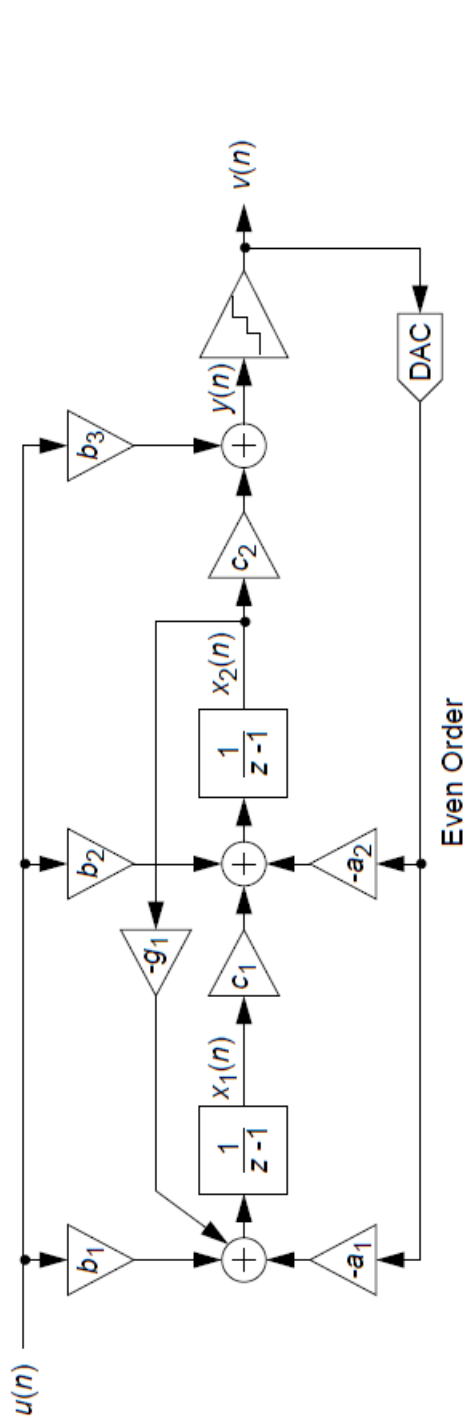
$$\begin{aligned} x(n+1) &= Ax(n) + B \begin{bmatrix} u(n) \\ v(n) \end{bmatrix} \\ y(n) &= Cx(n) + D \begin{bmatrix} u(n) \\ v(n) \end{bmatrix} \end{aligned} \tag{2}$$

この方程式は、線形なループフィルタを持つ全ての単一量子化器構造において十分に一般的である。本ツールボックスでは以下の構造に関して係数とABCD記述の間の変換が可能である。

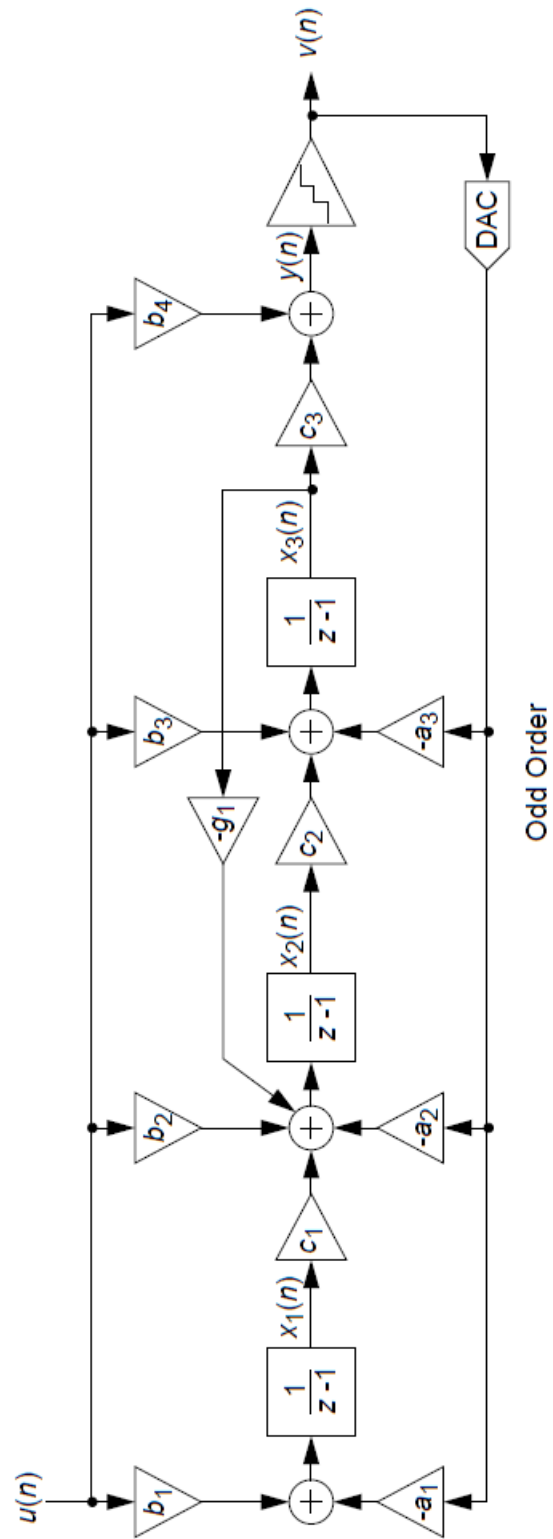
- CIFB Cascade-of-integrators, feedback form.
- CIFF Cascade-of-integrators, feedforward form.
- CRFB Cascade-of-resonators, feedback form.
- CRFF Cascade-of-resonators, feedforward form.

多入力で複数の量子化器を持つシステムもABCD行列で記述でき式2も利用できる。ni-入力, no-出力の変調器はA: $n \cdot n$, B: $n \cdot (ni+no)$, C: $no \cdot n$ で D: $no \cdot (ni+no)$ の行列で記述可能である。

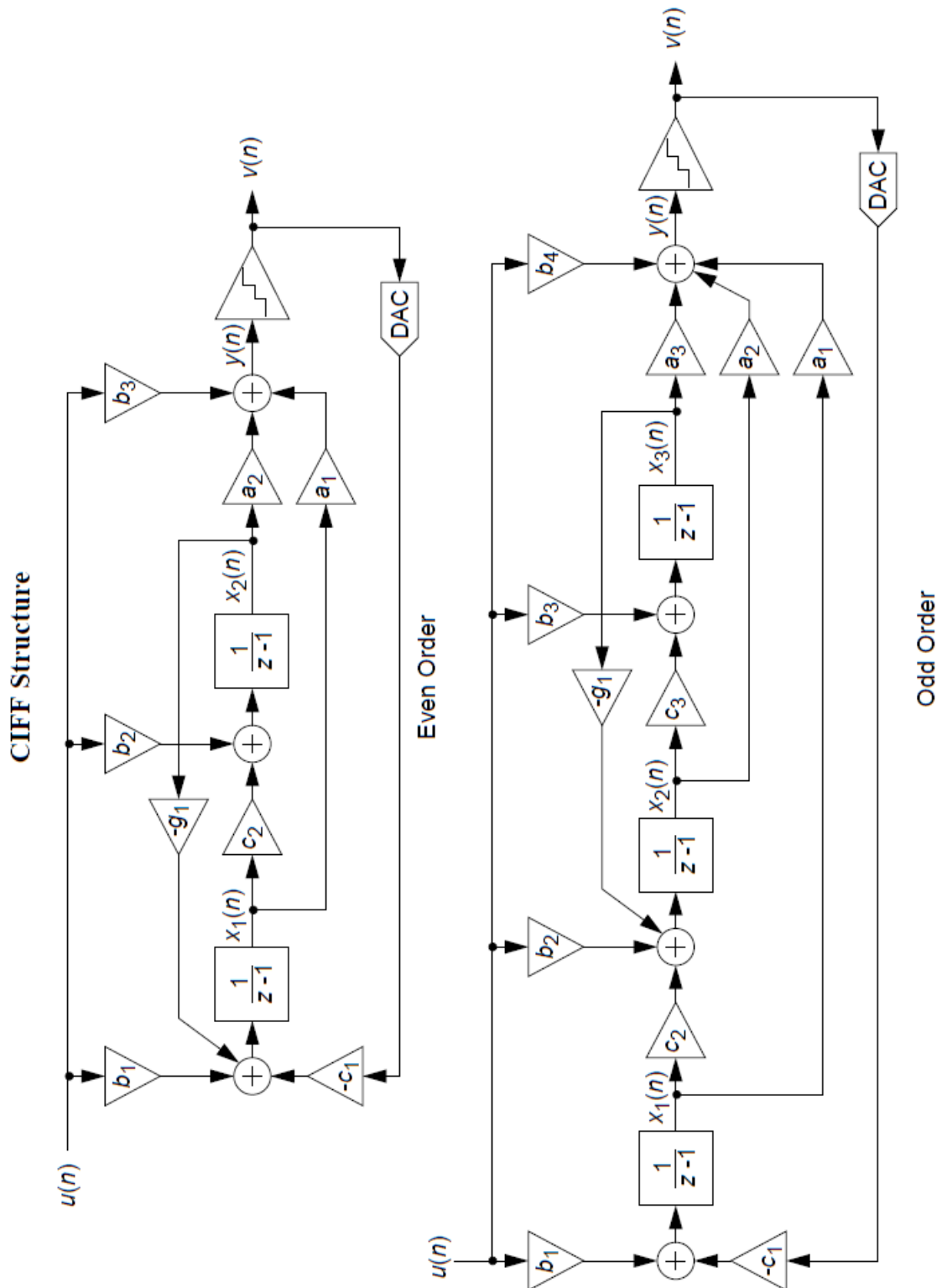
CIFB Structure

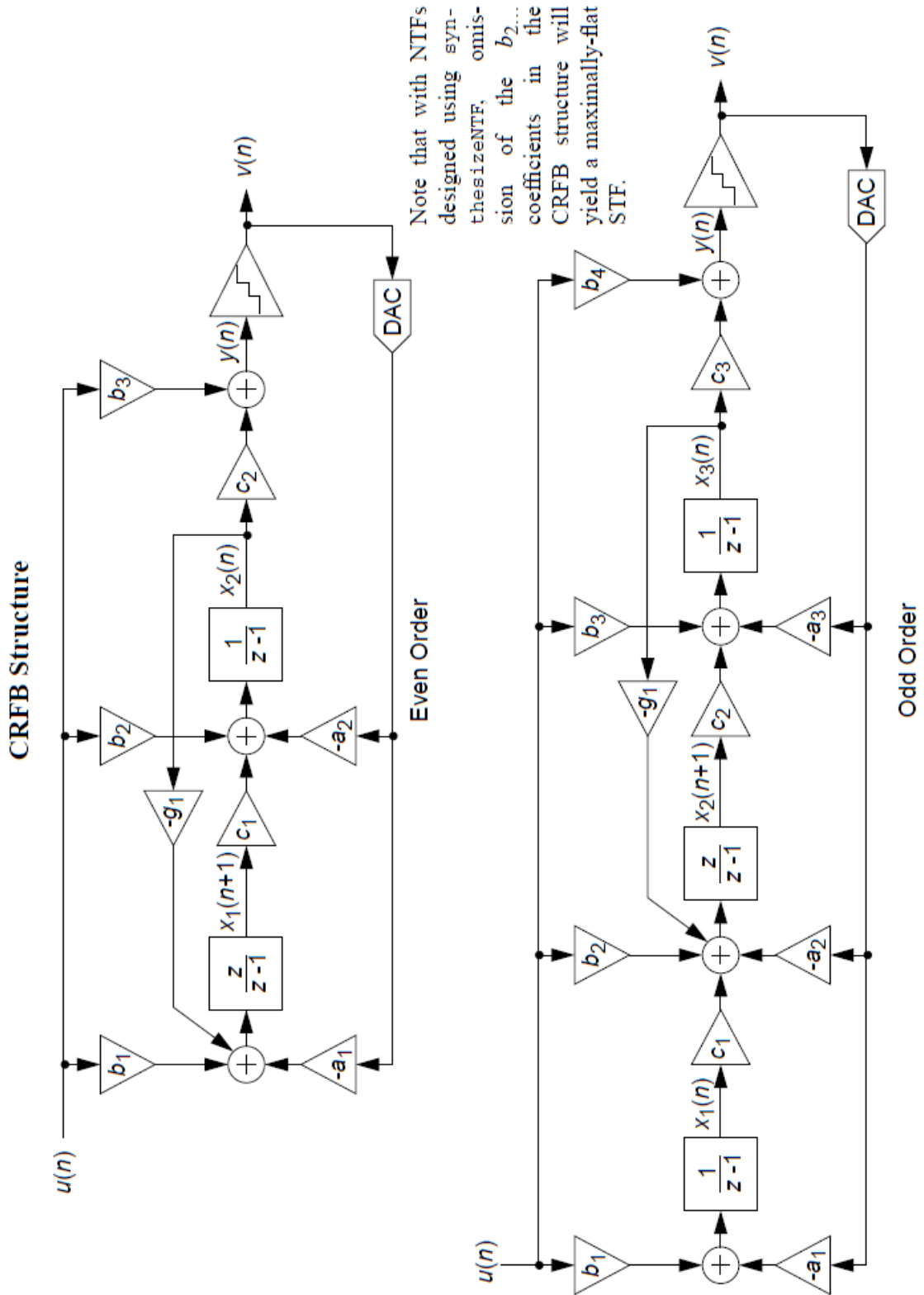


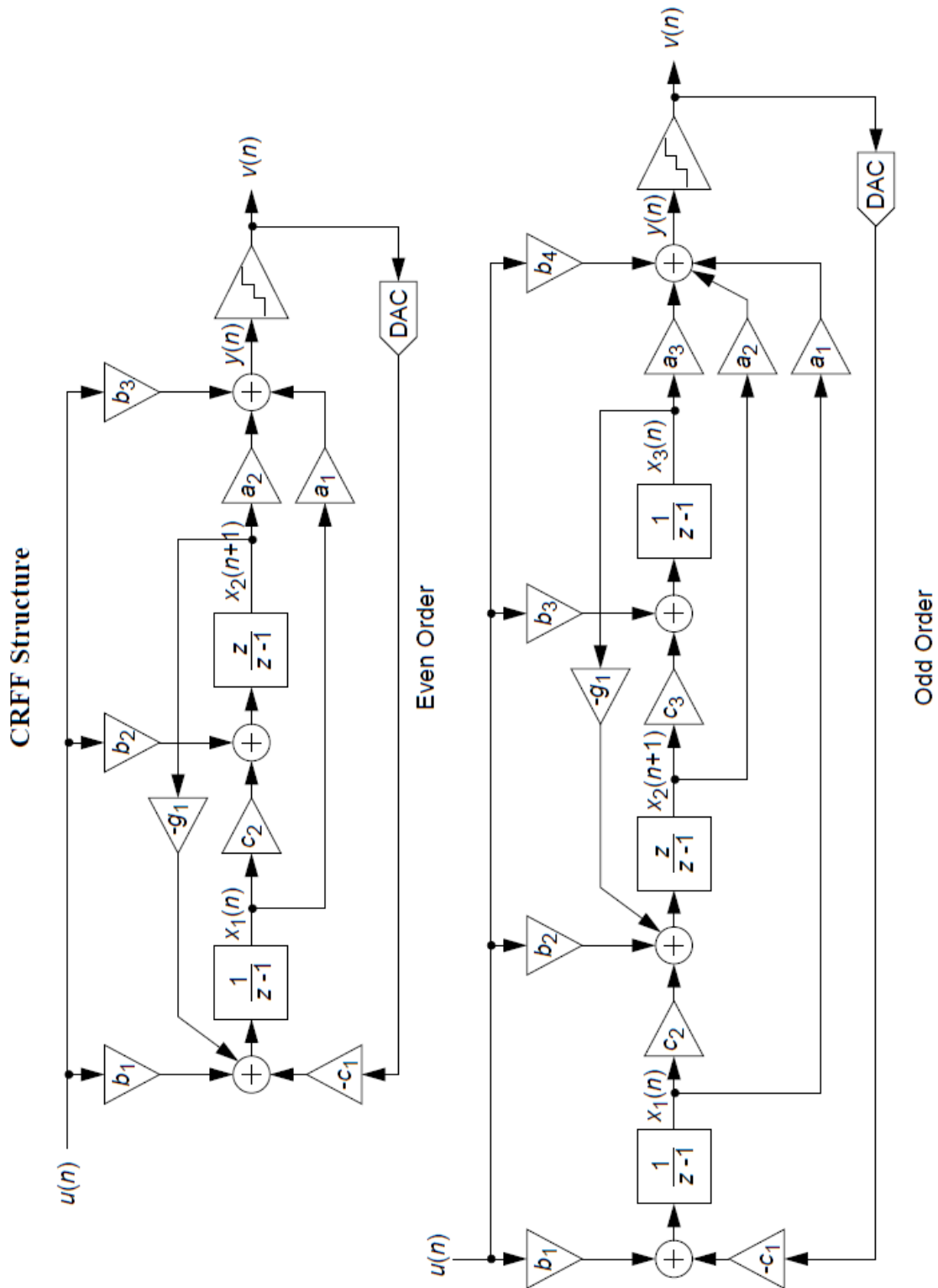
Even Order



Odd Order

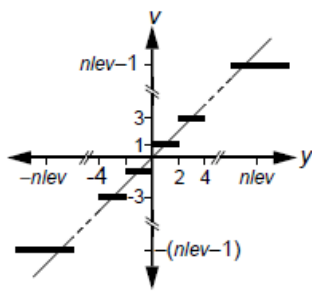




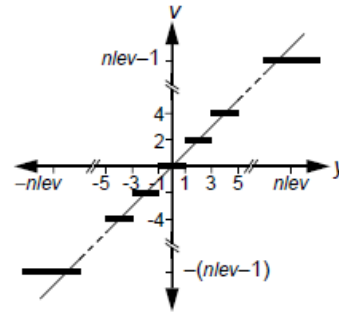


量子化器

量子化器は理想的であり、出力の中心値は0である。偶数のレベルを持つ量子化器は奇数の出力値を出力するミッドライズ型であり、奇数のレベルを持つ量子化器は偶数の出力値を出力するミッドトレッド型である。



偶数のレベルを持つ量子化器の伝達カーブ。



奇数のレベルを持つ量子化器の伝達カーブ。

参考図書

「ΔΣ型アナログ/デジタル変換器入門」

監訳者 和保 孝夫
安田 彰

発行所 丸善株式会社
発行 平成19年8月15日

ISBN 978-4-621-07872-3 C3055

※本マニュアルの著作権も丸善株式会社に帰属します。

株式会社 Trigence Semiconductor

高精度アナログ/デジタル回路設計の専門会社。

URL <http://www.trigence.co.jp>

代表 安田 彰