The Delta-Sigma Toolbox

Getting Started
To obtain your free copy of the Delta-Sigma Toolbox, go to the MathWorks web site (http://www.mathworks.com/matlabcentral/fileexchange), find the Controls category and select delsig. To improve simulation speed, compile the simulateDSM.c file by typing mex simulateDSM.c at the MATLAB prompt. Do the same for simulateDSL.c and al2m1f.c. The Delta-Sigma Toolbox requires the Signal Processing Toolbox and the Control Systems Toolbox; the clans function also requires the Optimization Toolbox. These and other toolboxes may be purchased from the MathWorks.

Toolbox Conventions
Frequencies are normalized; \( f = 1 \) corresponds to the sampling frequency \( (f_s) \).
Default values for function arguments are shown following an equals sign (=) in the parameter list. To use the default value for an argument, omit the argument if it is at the end of the list, otherwise use NaN (not-a-number) or {} (the empty matrix) as a placeholder.
A matrix is used to describe the loop filter of a general single-quantizer delta-sigma modulator. See “MODULATOR MODEL DETAILS” on page 410, for a description of this \( ABCD \) matrix.
DEMONSTRATIONS AND EXAMPLES

dsdemo1 Demonstration of the `synthesizeNTF` function. Noise transfer function synthesis for a 5th-order lowpass modulator, both with and without optimized zeros, plus an 8th-order bandpass modulator with optimized zeros.

dsdemo2 Demonstration of the `simulateDSM`, `predictSNR` and `simulateSNR` functions: time-domain simulation, SNR prediction using the describing function method of Ardalan and Paulos, spectral analysis and signal-to-noise ratio. Lowpass, bandpass, multi-bit lowpass examples are given.

dsdemo3 Demonstration of the `realizeNTF`, `stuffABCD`, `scaleABCD` and `mapABCD` functions: coefficient calculation and dynamic range scaling.

dsdemo4 Audio demonstration of MOD1 and MOD2 with sinc⁶ decimation.

dsdemo5 Demonstration of the `simulateESL` function: simulation of the element selection logic of a mismatch-shaping DAC.

dsdemo6 Demonstration of the `designHBF` function. Hardware-efficient halfband filter design and simulation.

dsdemo7 Demonstration of the `findFIS` function: positively-invariant set computation.

dsexample1 Discrete-time lowpass modulator design example.

dsexample2 Discrete-time bandpass modulator design example.
SUMMARY OF KEY FUNCTIONS

ntf = synthesizeNTF(order=3, OSR=64, opt=0, H_inf=1.5, f0=0)  page 393
ntf = clans(order=4, OSR=64, Q=5, rmax=0.95, opt=0)  page 394

Sythesize a noise transfer function.

[snr, amp, k0, k1, sigma_e2] = predictSNR(ntf, OSR=64, amp=..., f0=0)  page 395
Predict the SNR vs. input power curve using the describing function method
of Ardalan and Paulos.

[v, x, xmax, y] = simulateDSM(u, ABCD, nlev=2, x0=0)  page 396
[v, x, xmax, y] = simulateDSM(u, ntf, nlev=2, x0=0)
Simulate a delta-sigma modulator with a given input.

[snr, amp] = simulateSNR(ntf, OSR, amp=..., f0=0, nlev=2, f=1/(4*R), k=13)  page 398
Determine the SNR vs. input power curve by simulation.

[a, g, b, c] = realizeNTF(ntf, form='CRFB', stf=1)  page 400
Convert a noise transfer function into coefficients for a specific structure.

ABCD = stuffABCD(a, g, b, c, form='CRFB')  page 401
Calculate the ABCD matrix given the parameters of a specified modulator
topology.

[a, g, b, c] = mapABCD(ABCD, form='CRFB')  page 401
Calculate the parameters of a specified modulator topology given the ABCD
matrix.

[ABCDs, umax] = scaleABCD(ABCD, nlev=2, f=0, xlim=1, ymax=nlev+2)  page 402
Perform dynamic range scaling on a delta-sigma modulator described by
ABCD.

[ntf, stf] = calculateNTF(ABCD, k=1)  page 403
Calculate the NTF and STF of a delta-sigma modulator described by the
ABCD matrix, assuming a quantizer gain of k.

[s, x, sigma_se, max_sx, max_y] = simulateSSL(v, ntf, M=10, dw=[1...], sx0=[0...])  page 404
Simulate the element-selection logic in a mismatch-shaping DAC.

[f1, f2, info] = designHBF(fp=0.2, delta=1e-5, debug=0)  page 405
Design a hardware-efficient half-band filter for use in a decimation or inter-
polation filter.

y = simulateHBF(x, f1, f2, mode=0)  page 408
Simulate a Saramäki half-band filter in the time domain.

[s, e, n, o, s] = findPIS(u, ABCD, nlev=2, options)  page 409
Find a convex positively-invariant set for a delta-sigma modulator.
SUMMARY OF OTHER SELECTED FUNCTIONS

Delta-Sigma Utility

mod1, mod2
   Scripts for setting up the ABCD matrix, NTF and STF of the 1st/2nd-order
   modulator.

snr = calculateSNR(hwfft, f)
   Estimate the SNR given the in-band bins of a Hann-windowed FFT and the
   location of the input signal.

[sys, Gp] = mapCtoD(sys_c, t=[0 1], f0=0)
   Map a continuous-time system to a discrete-time system whose impulse
   response matches the sampled pulse response of the original continuous-
   time system.

[A B C D] = partitionABCD(ABCD, m)
   Partition ABCD into A, B, C, D for an m-input state-space system.

H_inf = infnorm(H)
   Compute the infinity norm (maximum absolute value) of a z-domain trans-
   fer function. See evalTF.

sigma_H = rmsGain(H, f1, f2)
   Compute the root mean-square gain of the discrete-time transfer function H
   in the frequency band (f1, f2).

General Utility

dbv(), dbp(), undbv(), undbp(), dbm()
   The dB equivalent of voltage/power quantities, and their inverse functions.

window = hann(N)
   A Hann window of length N. Unlike MATLAB's original hanning function,
   hann does not smear a tone which is located exactly in an FFT bin (i.e. the
   tone has an integer number of cycles in the given block of data). MATLAB
   6's hanning(N, 'periodic') function is the same as hann(N).

Graphing

plotPZ(H, color='b', markersize=5, list=0)
   Plot the poles and zeros of a transfer function.

figureMagic(xRange, dx, xLab, yRange, dy, yLab, size)
   Performs a number of formatting operations for the current figure, including
   axis limits, ticks and labelling.

printmif(file, size, font, fig)
   Print graph to an Adobe Illustrator file and then use ai2mif to convert it to
   FrameMaker MIF format. ai2mif is an improved version of the function of
   the same name originally written by Deron Jackson <djackson@mit.edu>.

[f, p] = logsMOOTH(X, inBin, nbin)
   Smooth the FFT, X, and convert it to dB. See also blgsmooth and bilog-
   plot for bandpass modulators.

**synthesizeNTF**

**Synopsis:** \( ntf = \text{synthesizeNTF}(\text{order}=3, \text{OSR}=64, \text{opt}=0, H_{\text{inf}}=1.5, f_0=0) \)

Synthesize a noise transfer function (NTF) for a delta-sigma modulator.

**Arguments**

- **order**
  The order of the NTF. \( \text{order} \) must be even for bandpass modulators.

- **OSR**
  The oversampling ratio. \( \text{OSR} \) is only needed when optimized NTF zeros are requested.

- **opt**
  A flag used to request optimized NTF zeros. \( \text{opt}=0 \) puts all NTF zeros at band-center (DC for lowpass modulators). \( \text{opt}=1 \) optimizes the NTF zeros. For even-order modulators, \( \text{opt}=2 \) puts two zeros at band-center, but optimizes the rest.

- **\( H_{\text{inf}} \)**
  The maximum out-of-band gain of the NTF. Lee’s rule states that \( H_{\text{inf}} \leq 2 \) should yield a stable modulator with a binary quantizer.
  Reducing \( H_{\text{inf}} \) increases the likelihood of success, but reduces the magnitude of the attenuation provided by the NTF and thus the theoretical resolution of the modulator.

- **\( f_0 \)**
  The center frequency of the modulator. \( f_0 
eq 0 \) yields a bandpass modulator; \( f_0=0.25 \) puts the center frequency at \( f_s/4 \).

**Output**

- **ntf**
  The modulator NTF, given as an LTI object in zero-pole form.

**Example**

Fifth-order lowpass modulator; zeros optimized for an oversampling ratio of 32.

\[ H = \text{synthesizeNTF}(5, 32, 1) \]

![Pole-Zero Plot](image1.png)

![Magnitude Response](image2.png)

Max. gain = 1.5 \( (3.5\, \text{dB}) \)

RMS gain = \(-65\, \text{dB}\)
clans

Synopsis:  \( ntf = \text{clans}(\text{order}=4, \text{OSR}=64, Q=5, rmax=0.95, \text{opt}=0) \)

Synthesize a noise transfer function (NTF) for a lowpass delta-sigma modulator using the
CLANS (Closed-loop analysis of noise-shaper) methodology [1]. This function requires
the optimization toolbox.


Arguments

- **order**  
The order of the NTF.
- **OSR**  
The oversampling ratio.
- **Q**  
The maximum number of quantization levels used by the fed-back
quantization noise. (Mathematically, \( Q = \|h\|_1 - 1 \), i.e. the sum of
the absolute values of the impulse response samples minus 1, is the
maximum instantaneous noise gain.)
- **rmax**  
The maximum radius for the NTF poles.
- **opt**  
A flag used to request optimized NTF zeros. \( \text{opt}=0 \) puts all NTF zeros at
band-center (DC for lowpass modulators). \( \text{opt}=1 \) optimizes the NTF
zeros. For even-order modulators, \( \text{opt}=2 \) puts two zeros at band-center,
but optimizes the rest.

Output

- **ntf**  
The modulator NTF, given as an LTI object in zero-pole form.

Example

5th-order lowpass modulator; time-domain noise gain of 5, zeros optimized for \( \text{OSR} = 32 \).
\[ n = \text{clans}(5, 32, 5, .95, 1) \]
predictSNR

Synopsis:  \( \text{predictSNR}(\text{ntf}, \text{OSR}=64, \text{amp}=\ldots, f0=0) \)
Use the describing function method of Ardalan and Paulos [1] to predict the signal-to-noise ratio (SNR) in dB for various input amplitudes. This method is only applicable to binary modulators.


Arguments
- **ntf**: The modulator NTF, given in zero-pole form.
- **OSR**: The oversampling ratio. OSR is used to define the "band of interest."
- **amp**: A row vector listing the amplitudes to use. Defaults to [-120 -110 -20 -15 -10 -9 -8 ... 0] dB, where 0 dB means a full-scale sine wave.
- **f0**: The center frequency of the modulator. \( f0 \neq 0 \) corresponds to a bandpass modulator.

Output
- **snr**: A row vector containing the predicted SNRs.
- **amp**: A row vector listing the amplitudes used.
- **k0**: The signal gain of the quantizer model; one value per input level.
- **k1**: The noise gain of the quantizer model; one value per input level.
- **sigma_e2**: The mean square value of the noise in the model of the quantizer.

Example
See the example on page 399.

The Quantizer Model:
The binary quantizer is modeled as a pair of linear gains and a noise source, as shown in the figure below. The input to the quantizer is divided into signal and noise components which are processed by signal-dependent gains \( k_0 \) and \( k_1 \). These signals are added to a noise source, which is assumed to be white and to have a Gaussian distribution (the variance \( \sigma_e^2 \) is also signal-dependent), to produce the quantizer output.

\[ y \rightarrow \text{sgn()} \rightarrow v \]  \( \rightarrow \)  \( y \rightarrow k_0 \)  \( y_0 \)  \( + \)  \( k_1 \)  \( y_1 \)  \( e: \text{AWGN with variance } \sigma_e^2 \)  \( v \)
simulateDSM

Synopsis:  \[ [v, x_n, x_{max}, y] = \text{simulateDSM}(u, \text{ABCD}, n\text{lev}=2, x0=0) \text{ or} \]
\[ [v, x_n, x_{max}, y] = \text{simulateDSM}(u, \text{ntf}, n\text{lev}=2, x0=0) \]

Simulate a delta-sigma modulator with a given input. For maximum speed, make sure that
the mex file is on your search path (At the MATLAB prompt, type which simulateDSM).

Arguments
\[ u \]
The input sequence to the modulator, given as a \( m \times N \) row vector. \( m \) is
the number of inputs (usually 1). Full-scale corresponds to an input of
magnitude \( n\text{lev}=1 \).

\[ \text{ABCD} \]
A state-space description of the modulator loop filter.

\[ \text{ntf} \]
The modulator NTF, given in zero-pole form.
The modulator STF is assumed to be unity.

\[ n\text{lev} \]
The number of levels in the quantizer. Multiple quantizers are indicated
by making \( n\text{lev} \) an array.

\[ x0 \]
The initial state of the modulator.

Output
\[ v \]
The samples of the output of the modulator, one for each input sample.

\[ x_n \]
The internal states of the modulator, one for each input sample, given as
an \( n \times N \) matrix.

\[ x_{max} \]
The maximum absolute values of each state variable.

\[ y \]
The samples of the quantizer input, one per input sample.
Example
Simulate a 5th-order binary modulator with a half-scale sine-wave input and plot its output in the time and frequency domains.

\[
\text{OSR} = 32;
\text{N} = \text{synthesizeNTF}(5, \text{OSR}, 1);
\text{fB} = \text{ceil}(\text{N} / (2 * \text{OSR}));
\text{f} = 85;
\text{u} = 0.5 * \sin(2 * \pi * \text{f} / \text{N} * [0: \text{N} - 1]);
\text{v} = \text{simulateDSM}(\text{u}, \text{N});
\]

Time-domain plot:
\[
\text{t} = 0:85;
\text{stairs}(\text{t}, \text{u}(\text{t}+1));
\text{hold on};
\text{stairs}(\text{t}, \text{v}(\text{t}+1));
\text{axis}([0 85 -1.2 1.2]);
\text{ylabel}('u, v');
\]

Frequency-domain plot:
\[
\text{spec} = \text{fft}(\text{v} * \text{hann}(	ext{N}))/\left(\text{N}/4\right);
\text{plot(lin}}\text{space}([0, 1, \text{N}/2]), \text{dbv(spec(1:} \text{N}/2));
\text{figureMagic([0 0.5], 0.05, 2, [-120 0], 10, 4);}
\text{ylabel('DB');}
\text{snr} = \text{calculateSNR(spec(1:} \text{fB}), \text{f});
\text{s} = \text{sprintf('SNR = %g.1dB', snr)};
\text{text(0.25, -90, s);}
\text{s} = \text{sprintf('NBW=%.15f', 1.5/\text{N});}
\text{text(0.25, -110, s);}
\]

SNR = 82.5dB
NBW=0.00018
simulateSNR

Synopsis:  \[ \text{[snr, amp]} = \text{simulateSNR}(\text{ntf, OSR, amp, } f0=0, nlev=2, f=1/} \]
\[ (4*\text{OSR}), k=13) \]

Simulate a delta-sigma modulator with sine wave inputs of various amplitudes and calculate the signal-to-noise ratio (SNR) in dB for each input.

Arguments

\textit{ntf}  
The modulator NTF, given in zero-pole form.

\textit{OSR}  
The oversampling ratio. OSR is used to define the “band of interest.”

\textit{amp}  
A row vector listing the amplitudes to use. Defaults to \([-120 -110 ... -20 -15 -10 -9 -8 ... 0] \text{ dB}, where 0 \text{ dB means a full-scale sine wave, i.e. a sine wave whose peak value is } nlev=1 \).

\textit{f0}  
The center frequency of the modulator. \(f0\neq0\) corresponds to a bandpass modulator.

\textit{nlev}  
The number of levels in the quantizer.

\textit{f}  
The normalized frequency of the test sinusoid; a check is made that the test frequency is in the band of interest. The frequency is also adjusted so that it lies precisely in an \texttt{FFT} bin.

\textit{k}  
The number of time points used for the \texttt{FFT} is \(2^k\).

Output

\textit{snr}  
A row vector containing the SNR values calculated from the simulations.

\textit{amp}  
A row vector listing the amplitudes used.
Example
Compare the SNR vs. input amplitude curve for a fifth-order modulator determined by the
describing function method with that determined by simulation.

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');
figureMagic([-100 0], 10, 1, [0 100], 10, 1);
xlabel('Input Level, dB');
ylabel('SNR dB');
title('SNR curve');
s=sprintf('peak SNR = %4.1f dB\n', max(snr));
text(-49,15,s);

SNR curve
realizeNTF

Synopsis: \[ [a, g, b, c] = \text{realizeNTF}(\text{ntf}, \text{form} = \text{'CRFB'}, \text{stf} = 1) \]

Convert a noise transfer function (NTF) into a set of coefficients for a particular modulator topology.

Arguments

\textit{ntf}  
The modulator NTF, given in zero-pole form (i.e. a \texttt{zpk} object).

\textit{form}  
A string specifying the modulator topology.  
\begin{itemize}
  \item \text{CRFB}  
  Cascade-of-resonators, feedback form.
  \item \text{CRFF}  
  Cascade-of-resonators, feedforward form.
  \item \text{CIFB}  
  Cascade-of-integrators, feedback form.
  \item \text{CIFF}  
  Cascade-of-integrators, feedforward form.
\end{itemize}

Structures are described in detail in "MODULATOR MODEL DETAILS" on page 410.

\textit{stf}  
The modulator STF, specified as a \texttt{zpk} object. Note that the poles of the STF must match those of the NTF in order to guarantee that the STF can be realized without the addition of extra state variables.

Output

\textit{a}  
Feedback/feedforward coefficients from/to the quantizer \((1 \times n)\).

\textit{g}  
Resonator coefficients \((1 \times \lfloor n/2 \rfloor)\).

\textit{b}  
Feed-in coefficients from the modulator input to each integrator \((1 \times (n + 1))\).

\textit{c}  
Integrator inter-stage coefficients. \((1 \times n\). In unscaled modulators, \textit{c} is all ones.)

Example

Determine the coefficients for a 5\textsuperscript{th}-order modulator with the cascade-of-resonators structure, feedback (CRFB) form.

\begin{verbatim}
>> H = \text{synthesizeNTF}(5, 32, 1);
>> [a, g, b, c] = \text{realizeNTF}(H, \text{'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
\end{verbatim}
stuffABCD

**Synopsis:** $ABCD = \text{stuffABCD}(a,g,b,c,\text{form='CRFB'})$
Calculate the ABCD matrix given the parameters of a specified modulator topology.

**Arguments**
- $a$: Feedback/feedforward coefficients from/to the quantizer. $1 \times n$
- $g$: Resonator coefficients. $1 \times \lfloor n/2 \rfloor$
- $b$: Feed-in coefficients from the modulator input to the input of each integrator and to the input of the quantizer. $1 \times n + 1$
- $c$: Integrator inter-stage coefficients. $1 \times n$
- $\text{form}$: see `realizeNTP` on page 400 for a list of supported structures.

**Output**
$ABCD$: A state-space description of the modulator loop filter.

mapABCD

$\{a,g,b,c\} = \text{mapABCD}(ABCD,\text{form='CRFB'})$
Calculate the parameters for a specified modulator topology, assuming ABCD fits that topology.

**Arguments**
- $ABCD$: A state-space description of the modulator loop filter.
- $\text{form}$: see `realizeNTP` on page 400 for a list of supported structures.

**Output**
- $a$: Feedback/feedforward coefficients from/to the quantizer. $1 \times n$
- $g$: Resonator coefficients. $1 \times \lfloor n/2 \rfloor$
- $b$: Feed-in coefficients from the modulator input to each integrator. $1 \times n + 1$
- $c$: Integrator inter-stage coefficients. $1 \times n$
scaleABCD

Synopsis: \( [ABCDs, umax] = \text{scaleABCD} (ABCD, nlev=2, f=0, xlim=1, ymax=nlev+5, umax, N=1e5) \)

Scale the ABCD matrix so that the state maxima are less than a specified limit. The maximum stable input is determined as a side-effect of this process.

Arguments

- **ABCD**: A state-space description of the modulator loop filter.
- **nlev**: The number of levels in the quantizer.
- **f**: The normalized frequency of the test sinusoid.
- **xlim**: The limit on the states. May be given as a vector.
- **ymax**: The threshold for judging modulator stability. If the quantizer input exceeds ymax, the modulator is considered to be unstable.
- **umax**: The maximum input which will be applied to the modulator during simulation. If umax is not supplied, it will be estimated by simulation.
- **N**: The number of time steps used for determination of the state maxima.

Output

- **ABCDs**: The scaled state-space description of the modulator loop filter.
- **umax**: The maximum stable input. Input sinusoids with amplitudes below this value should not cause the modulator states to exceed their specified limits.
calculateTF

Synopsis:  [ntf, stf] = calculateTF(ABCD, k=1)
Calculate the NTF and STF of a delta-sigma modulator.

Arguments
ABCD  A state-space description of the modulator loop filter.
k  The value to use for the quantizer gain.

Output
ntf  The modulator NTF, given as an LTI system in zero-pole form.
stf  The modulator STF, given as an LTI system in zero-pole form.

Example
Realize a fifth-order modulator with the cascade-of-resonators structure, feedback form. Calculate the ABCD matrix of the loop filter and verify that the NTF and STF are correct.

```matlab
>> H = synthesizeNTF(5,32,1)
Zero/pole/gain:
(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)
Sampling time: 1
>> [a,g,b,c] = realizeNTF(H)
a =
0.0007   0.0084   0.0550   0.2443   0.5579
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)
Zero/pole/gain:
(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)
Sampling time: 1
Zero/pole/gain:
1
Static gain.
```
simulateESL

Synopsis: \[ sv, sx, sigma\_se, max\_sx, max\_sy \]
\[
= \text{simulateESL}(v, mtf, M=16, dw=[1...], sx0=[0...])
\]
Simulate the element selection logic (ESL) of a multi-element DAC using a particular mismatch-shaping transfer function \( mtf \).


Arguments

\( v \) 
A vector containing the number of elements to enable. Note that the output of \text{simulateDSM} must be offset and scaled in order to be used here as \( v \) must be in the range \( [0, \sum M dw(i)] \).

\( mtf \) 
The mismatch-shaping transfer function, given in zero-pole form.

\( M \) 
The number of elements.

\( dw \) 
A vector containing the weight associated with each element.

\( sx0 \) 
An \( n \times M \) matrix containing the initial state of the element selection logic.

Output

\( sv \) 
The selection vector: a vector of zeros and ones indicating which elements to enable.

\( sx \) 
An \( n \times M \) matrix containing the final state of the element selection logic.

\( sigma\_se \) 
The rms value of the selection error, \( se = sv - sy \). \( sigma\_se \) may be used to analytically estimate the power of in-band noise caused by element mismatch.

\( max\_sx \) 
The maximum value attained by any state in the ESL.

\( max\_sy \) 
The maximum value attained by any component of the (un-normalized) "desired usage" vector.

Example

Run \text{dsdemo6.m}.
designHBF

Synopsis: \([f1,f2,info]=\text{designHBF}(fp=0.2, \text{delta}=1e-5, \text{debug}=0)\)
Design a hardware-efficient linear-phase half-band filter for use in the decimation or interpolation filter associated with a delta-sigma modulator. This function is based on the procedure described by Saramäki [1]. Note that since the algorithm uses a non-deterministic search procedure, successive calls may yield different designs.


Arguments
- **fp**
  - Normalized passband cutoff frequency.
- **delta**
  - Passband and stopband ripple in absolute value.

Output
- **f1,f2**
  - Prototype filter and subfilter coefficients and their canonical-signed digit (csd) representation.
- **info**
  - A vector containing the following information data (only set when \(\text{debug}=1\)):
    - complexity
      - The number of additions per output sample.
    - n1,n2
      - The length of the \(f1\) and \(f2\) vectors.
    - sbr
      - The achieved stop-band attenuation (dB).
    - phi
      - The scaling factor for the \(f2\) filter.

Example
Design of a lowpass half-band filter with a cut-off frequency of \(0.2f_s\), a passband ripple of less than \(10^{-5}\) and a stopband rejection of at least \(10^{-5}\) (-100 dB).

```matlab
[f1,f2] = \text{designHBF}(0.2, 1e-5);
f = \text{linspace}(0, 0.5, 1024);
plot(f, \text{dbv}(\text{freqHBF}(f, f1, f2)))
```

A plot of the filter response and the structure of this filter as a decimation or as an interpolation filter are shown on the next page. The filter achieves 109 dB of attenuation in the stopband and uses only 124 additions (no true multiplications) to produce each output sample.
The coefficients and their signed-digit decompositions are

\[
\begin{align*}
[f1\_val]' = [f2\_val]' & >> f1\_csd >> f2\_csd \\
0.9453 & 0.6211 & \text{ans} = 0 & -4 & -7 & -1 & -3 & -8 \\
-0.6406 & -0.1895 & 1 & -1 & 1 & \text{ans} = & -2 & -4 & -9 \\
0.1953 & 0.0957 & \text{ans} = & -1 & -3 & -6 & -1 & 1 & -1 \\
-0.0508 & 0.0269 & \text{ans} = & -1 & -1 & -1 & \text{ans} = & -3 & -5 & -9 \\
0.0142 & \text{ans} = & -2 & -4 & -7 & -1 & -1 & 1 & 1 \\
\end{align*}
\]

In the above signed-digit expansions, the first row contain the powers of two while the second row gives their signs. For example, \(f_1(1) = 0.9453 = 2^0 - 2^{-4} + 2^{-7}\) and \(f_2(1) = 0.6211 = 2^{-1} + 2^{-3} - 2^{-8}\). Since the filter coefficients for this example use only 3 signed digits, each multiply-accumulate operation shown in the diagram below needs only 3 binary additions. Thus, an implementation of this 110th-order FIR filter needs to perform only \(3 \times 3 + 5 \times (3 \times 6 + 6 - 1) = 124\) additions at the low \((f_s/2)\) rate.
**simulateHBF**

**Synopsis:**  \( y = \text{simulateHBF}(x,f1,f2,\text{mode}=0) \)  
Simulate a Saramäki half-band filter (see designHBF on page 405) in the time domain.

**Arguments**

- \( x \)  
  The input data.

- \( f1,f2 \)  
  Filter coefficients. \( f1 \) and \( f2 \) can be vectors of values or struct arrays like those returned from designHBF.

- \( \text{mode} \)  
  The mode flag determines whether the input is filtered, interpolated, or decimated according to the following:

  0   Plain filtering, no interpolation or decimation.

  1   The input is interpolated.

  2   The output is decimated, even samples are taken.

  3   The output is decimated, odd samples are taken.

**Output**

- \( y \)  
  The output data.

**Example**

Plot the impulse response of the HBF designed on the previous page.

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

![Impulse Response Graph](image)

**Sample Number**

**Impulse Response**
findPIS, find2dPIS (in the PosInvSet subdirectory)

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

Find a convex positively-invariant set for a delta-sigma modulator. findPIS requires compilation of the qhull mex file; find2dPIS does not, but is limited to second-order systems.

Arguments

\( u \)
The input to the modulator. If \( u \) is a scalar, the input to the modulator is constant. If \( u \) is a \( 2 \times 1 \) vector, the input to the modulator may be any sequence whose samples lie in the range \( [u(1), u(2)] \).

\( ABCD \)
A state-space description of the modulator loop filter.

\( nlev \)
The number of quantizer levels.

\( dbg \)
Set \( dbg=1 \) to get a graphical display of the iterations.

\( itnLimit \)
The maximum number of iterations.

\( expFactor \)
The expansion factor applied to the hull before every mapping operation. Increasing \( expFactor \) decreases the number of iterations but results in sets which are larger than they need to be.

\( N \)
The number of points to use when constructing the initial guess.

\( skip \)
The number of time steps to run the modulator before observing the state. This handles the possibility of “transients” in the modulator.

\( \text{qhullArgA} \)
The 'A' argument to the qhull program. Adjacent facets are merged if the cosine of the angle between their normals is greater than the absolute value of this parameter. A negative value implies that the merge is performed during hull construction, rather than after.

\( \text{qhullArgC} \)
The 'C' argument to the qhull program. A facet is merged into its neighbor if the distance between the facet's centrum (the average of the facet's vertices) and the neighboring hyperplane is less than the absolute value of this parameter. As with \( \text{qhullArgA} \), a negative value implies pre-merging whereas a positive value implies post-merging.

Output

\( s \)
The vertices of the set \( (dim \times n_v) \).

\( e \)
The edges of the set, listed as pairs of vertex indices \( (2 \times n_v) \).

\( n \)
The normals for the facets of the set \( (dim \times n_f) \).

\( o \)
The offsets for the facets of the set \( (1 \times n_f) \).

\( Sc \)
The scaling matrix which was used internally to “round out” the set.
MODULATOR MODEL DETAILS

A delta-sigma modulator with a single quantizer is assumed to consist of quantizer connected to a loop filter as shown in the diagram below.

The Loop Filter
The loop filter is described by an ABCD matrix. For single-quantizer systems, the loop filter is a two-input, one-output linear system and ABCD is an $(n+1) \times (n+2)$ matrix, partitioned into $A (n \times n)$, $B (n \times 2)$, $C (1 \times n)$ and $D (1 \times 2)$ sub-matrices as shown below:

$$ABCD = \begin{bmatrix} A & B \\ C & D \end{bmatrix}.$$  \hspace{1cm} (B.1)

The equations for updating the state and computing the output of the loop filter are

$$x(n+1) = Ax(n) + Bu(n)$$
$$y(n) = Cx(n) + Du(n).$$  \hspace{1cm} (B.2)

This formulation is sufficiently general to encompass all single-quantizer modulators which employ linear loop filters. The toolbox currently supports translation from an ABCD description to coefficients and vice versa for the following topologies:

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

Multi-input and multi-quantizer systems are also described with an ABCD matrix and Eq. (2) still applies. For an $n^{th}$-order, $n_i$-input, $n_o$-output modulator, the dimensions of the sub-matrices are $A: n \times n$, $B: n \times (n_i+n_o)$, $C: n_o \times n$ and $D: n_o \times (n_i+n_o)$.
The Quantizer
The quantizer is as described in Section 2.1, namely a symmetric binary quantizer with a step size of two. Quantizers with an even number of levels are of the mid-rise type and produce outputs which are odd integers. Quantizers with an odd number of levels are of the mid-tread type and produce outputs which are even integers.

Transfer curve of a quantizer with an even number of levels.

Transfer curve of a quantizer with an odd number of levels.

Illustrations of the four topologies for which the toolbox supports coefficient calculation are given on the following pages.
Note that with NTFs designed using `synthesizeNTF`, omission of the $b_2$, coefficients in the CRFB structure will yield a maximally-flat STF.