ANSI FILTERING TOOLBOX

by Guillermo Jiménez-Arranz

The ANSI filtering toolbox provides the tools to design, visualise and apply a bank of bandpass filters that meet the ANSI S1.11 standard. Some features of this toolbox that makes it a better option that other available tools in MATLAB are:

- **The filter bank meets the ANSI S1.11 standard**.
- **It can cover any range of frequencies**, even outside the human audible region of 20 Hz to 20 kHz for which ANSI filter banks are typically designed.
- **The bandpass filters are not be affected by numerical instabilities**. The filtering methodology applies decimation to the signal to reduce its bandwidth and sampling frequency, thus simpler and stable bandpass filters can be applied.
- **It can be used to filter short, transient signals** down to frequencies where typical filters would cause serious issues due to delays, in particular when these are larger than the signal duration. The toolbox addresses this problem by padding the signal with a number of zeroes larger than several times the filter order or filter's group delay.
- **Zero-phase filtering** can be applied to avoid altering the phase information of the signal. This is particularly useful for calculating the peak amplitudes of the filtered signal.
- **The functions can output the filtered waveforms or the filtered metrics, per band**. The metrics include RMS, exposure, peak, and peak-to-peak.

The toolbox contains three main functions: `ansiFilterDesign`, `ansiFilter`, and `ansiFilterPlot`. The three are described in detail below.

`ansiFilterDesign`

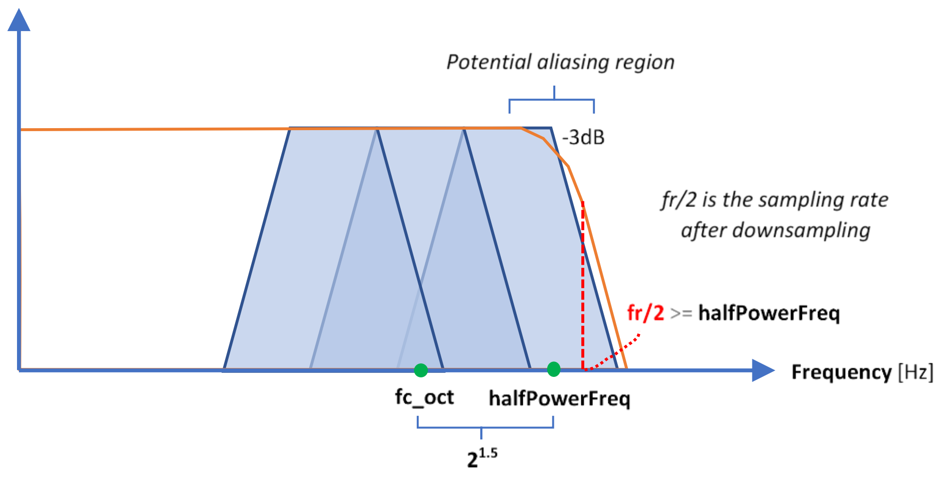
This function builds a set of decimation and bandpass filters based on a specified frequency range, filter order/class and bands per octave. A sequence of decimation stages will be applied to the signal, so that simpler and more effective bandpass filters can be designed. The decimation stages rely on a unique anti-aliasing filter. Similarly, a set of unique, low-order bandpass filters is designed to filter the signal after each decimation stage. The detailed filtering process is described in the following section. Below are described the fields contained in the output filter bank structure from `ansiFilterDesign`.

The design process for the filter bank is as follows:

1. Calculate the bottom, central and top frequencies (`f1`, `fc`, `f2`) for the fractional-octave bands within the specified frequency limits (`fmin`, `fmax`).
2. Calculate the parent octave bands (`f1_oct`, `fc_oct`, `f2_oct`) for the fractional-octave bands. The octave bands will be between `fmin/2` and `2*fmax`.
3. Calculate the indices of the parent octave bands corresponding to each fractional-octave band (`bpassOctIndex`). Remove any octaves with no associated fractional-octave bands.
4. Calculate the number of unique bandpass filters (`nUniqueFilters`)
5. Design the decimation filter (`decimFilter`). Note that in the anti-alias filter all normalised frequencies are referred to the sampling rate `fr` <u>before</u> downsampling. This makes sense, since

the alias filter is applied before downsampling. The decimation factor `D` always relates the original sample frequency `fs` to the sample frequency `fr` <u>after</u> downsampling.

- Find the central frequency of the first octave processed after decimation. This is the first octave that is at least $2^{2.5}$ times lower than `fs/2`.
- Calculate the half-power cutoff frequency of the decimator (`halfPowerFreq`). This is chosen to be $2^{1.5}$ times the central frequency of the octave band to be filtered (`halfPowerFreq = fc_oct` * $2^{1.5}$).
- Calculate the decimation factor (`D`). The decimation factor is calculated so that the Nyquist frequency <u>after</u> downsampling is higher or equal than the half-power cutoff frequency of the decimator (`fr/2 >= halfPowerFreq`).
- Calculate the sampling frequency <u>before</u> downsampling (`fr = 2*fs/D`).
- Calculate the normalised half-power cutoff frequency of the decimator <u>before</u> downsampling (`halfPowerFreqn = 2* halfPowerFreq/fr`).
- Design the anti-aliasing filter to be applied <u>before</u> downsampling. The normalised half-power cutoff frequency before downsampling (`halfPowerFreqn`) is used to design it.


Potential aliasing region
-3dB
fr/2 is the sampling rate after downsampling
fr/2 >= halfPowerFreq
Frequency [Hz]
fc_oct
halfPowerFreq
$2^{1.5}$

6. Design fractional octave band filters (`bpassFilter`). Note that in the bandpass filters all normalised frequencies are referred to the sampling rate `fr` <u>after</u> downsampling. This makes sense, since the bandpass filter is applied after decimation (anti-alias filtering + downsampling). The decimation factor `D` relates the original sample frequency `fs` to the sample frequency `fr` <u>after</u> downsampling. The actions below are applied to the current octave band after decimation.
   - Find the central frequency of the current octave.
   - Calculate the half-power cutoff frequency of the decimator (`halfPowerFreq`). This is chosen to be $2^{1.5}$ times the central frequency of the octave band.
   - Calculate the decimation factor (`D`). The decimation factor is calculated so that the Nyquist frequency <u>after</u> downsampling is higher or equal than the half-power cutoff frequency of the decimator (`fr/2 >= halfPowerFreq`).
   - Calculate sampling frequency <u>after</u> downsampling (`fr = fs/D`).

- Calculate the normalised half-power bottom and top cutoff frequencies of the bandpass filter <u>after</u> downsampling (`halfPowerFreqn1 = 2* halfPowerFreq1/fr`, and `halfPowerFreqn2 = 2* halfPowerFreq2/fr`).
- Design the bandpass filter for the current octave band.
- Repeat for the rest of fractional bands in the octave.
- Repeat for the rest of octaves until all the unique bandpass filters (`nUniqueFilters`) have been designed.

Field	**Parent Field**	**Description**
sampleRate	AnsiFilterBank	Sampling rate used to design the filter bank [Hz]
freqLimits	AnsiFilterBank	Bottom and top frequency limits of the filter bank [Hz]
freqLimitMode	AnsiFilterBank	Reference point in a band used to determine whether it falls within the specified frequency limits. This is an input parameter of `fractionalOctaveBands`.
octaveRatio	AnsiFilterBank	Ratio between two frequencies for the numerator to be considered an octave of the denominator. Two options: base-10 ($10^{3/10}$) and base-2 (2). Base-2 is used for consistency with a decimation factor multiple of 2.
decimatorBank	AnsiFilterBank	Substructure containing the filtering object and info on the unique decimator.
bpassBank	AnsiFilterBank	Substructure containing the filtering object and info on the fractional octave bandpass filters in the bank.
filterType	decimatorBank	'lowpass' for decimation filter
filterResponse	decimatorBank	'IIR' (no FIR filters)
filterOrder	decimatorBank	Order of decimator. Specified as input or calculated from given filter class. All filters have the same order.
filterObject	decimatorBank	Unique unconstrained, lowpass, IIR Butterworth filter of digitalFilter class applied to all decimation stages.
halfPowerFreqn	decimatorBank	Normalised half-power (-3 dB) frequency for the decimation filter, <u>before</u> downsampling.
decimationFactor	decimatorBank	Factors by which the original sample rate `fs` is reduced to obtain the current sample rate `fr` for the decimator <u>after</u> downsampling.
sampleRate	decimatorBank	Sample rate at each decimation stage <u>before</u> downsampling (`fr = 2*fs/D`)
halfPowerFreq	decimatorBank	Absolute half-power (-3 dB) frequency for the decimation filter at each decimation stage.
parentOctaveIndex	decimatorBank	Indices of parent octave bands
parentOctaveCentralFreq	decimatorBank	Central frequency of parent octave bands [Hz].
parentOctaveHalfPowerFreq1	decimatorBank	Low-edge frequency of parent octave bands [Hz]
parentOctaveHalfPowerFreq2	decimatorBank	High-edge frequency of parent octave bands [Hz]
filterType	bandPassBank	'bandpass' for the fractional octave filters

filterResponse	bandPassBank	'IIR' (no FIR filters)
bandsPerOctave	bandPassBank	Number of fractional bands in an octave
filterOrder	bandPassBank	Order of the unique bandpass filters. Specified as input or calculated from given filter class. All filters have the same order.
filterClass	bandPassBank	Class of each of unique bandpass filters.
filterObject	bandPassBank	Unique unconstrained, bandpass, IIR Butterworth filters of digitalFilter class, from high to low freq.
centralFreqn	bandPassBank	Normalised central frequencies for the unique bandpass filters, in descending order.
halfPowerFreqn1	bandPassBank	Normalised bottom half-power (-3 dB) frequencies of the bandpass filters after downsampling, in descending order.
halfPowerFreqn2	bandPassBank	Normalised top half-power (-3 dB) frequencies of the bandpass filters after downsampling, in descending order.
filterObjectIndex	bandPassBank	Indices of the unique filter object that each bandpass filter must use.
decimationFactor	bandPassBank	Factors by which the original sample rate FS is reduced to obtain the current sample rate FR after downsampling.
sampleRate	bandPassBank	Sample rate at each decimation stage after downsampling (`fr = fs/D`)
centralFreq	bandPassBank	Absolute central frequencies for the unique bandpass filters, in descending order
halfPowerFreq1	bandPassBank	Normalised bottom half-power (-3 dB) frequencies of bandpass filters in descend. order.
halfPowerFreq2	bandPassBank	Normalised top half-power (-3 dB) frequencies of bandpass filters in descending order.
parentOctaveIndex	bandPassBank	Indices of parent octave bands.
parentOctaveCentralFreq	bandPassBank	Central frequencies of parent octave bands [Hz].
parentOctaveHalfPowerFreq1	bandPassBank	Bottom frequencies of parent octave bands [Hz].
parentOctaveHalfPowerFreq2	bandPassBank	Top frequencies of parent octave bands [Hz].

`ansiFilter` and the Filtering Process

The `ansiFilter` function filters the input signal using the filter bank defined in input structure `AnsiFilterBank`. The filtering process follows the steps below:

1. Zero-pad the input signal (if `'ZeroPad' = true`). One-sided zero-padding (ZP) is applied if `'FilterMode' = 'filter'` is selected, and two-sided ZP otherwise.
2. If `'MetricsOutput' = 0,1`, then for each octave containing at least one fractional octave band:
    a. Decimate as many times as needed to reach the first (highest) octave band. Each decimation step is carried out by simply filtering the signal with the anti-alias filter and then downsampling by 2. For the first octave band, more than one decimation step may be needed, but only one is required for every new octave after the first one.
    b. For each fractional band within the octave:

i. Filter the decimated signal with the appropriate unique bandpass filter.
ii. Interpolate the bandpass filtered signal if `‘MetricsOutput’ = 0` is selected.
iii. Calculate the metrics (RMS, exposure, peak, peak-to-peak) from the bandpass filtered signal if `‘MetricsOutput’ = 1` is selected.

3. If `‘MetricsOutput’ = 2`, then calculate the RMS and exposure metrics per fractional octave band using the FFT.

Function	Priority	Description	Dependencies
ansiFilterDesign	1	Designs a bank of digital IIR bandpass filters following ANSI S1.11 standard. The function returns a structure AnsiFilterBank containing the filtering objects and information about the decimator and bandpass filters in the bank. AnsiFilterBank is used as input of ansiFilter to extract the bandpass filtered signals or their metrics (RMS, Exposure, Peak, Peak-to-Peak) from a given broadband signal. AnsiFilterBank is also used as input of AnsiFilterPlot to plot the response of the decimators and bandpass filters in the bank with the ANSI masks to evaluate the compliance of the bandpass filters with the ANSI S1.11 standard.	fractionalOctaveBands ansiFilterCompliance ansiFilterMask
> fractionalOctaveBands	2	Calculates the central and band-edge frequencies of the fractional octave bands that are within the selected frequency range. The calculations are based on the BS EN 61260-1:2014 standard (equivalent to ANSI S1.11:2014).	*None*
> ansiFilterCompliance	2	Returns the ANSI S1.11 class that the input bandpass filter is compliant with (0,1,2). The function returns -1 if the input filter does not meet the class 2 specification, the least restrictive.	ansiFilterMask
> ansiFilterMask	2	Calculates the maximum and minimum attenuation limits for an ANSI S1.11 bandpass filter of bandwidth 1/bpo octaves, central frequency fc, and class filtClass.	*None*
ansiFilter	1	Filters the input signal x with the filter bank ansiFilterBank generated with ansiFilterDesign. The function returns the filtered data xb and the frequencies of the fractional octave bands FC. Property 'MetricsOutput' = 0 assigns the bandpass-filtered signals to XB. The function can also return the band metrics (RMS, Exposure, Peak, Peak-to-Peak) calculated with the ANSI filter bank ('MetricsOutput' = 1) or FFT filtering ('MetricsOutput' = 2). The filtering function can be selected with 'FiltMode' (see filter and myfiltfilt).	ansiFilterDesign myfiltfilt
ansiFilterPlot	1	Plots the decimators (red) and bank of bandpass filters (magenta) defined in the input structure ansiFilterBank (see ansiFilterDesign), along with the masks of the three classes defined in the ANSI S1.11 standard. The diamond markers on the decimator curves highlight the central frequency of their parent octaves and half-power points.	ansiFilterDesign
ansiFilter_Test	1	Test the three 'MetricsOutput' options in ansiFilter and compared their results with and without zero padding. It becomes clear that is necessary for short signals and low frequency filtering.	ansiFilterDesign ansiFilter
compactnum	1	Converts the input numerical vector x into a matrix of alpha-numerical strings. Each string consists of the number expressed as floating point with ndec decimal places followed by a metric prefix letter. For example, for x = 2525 and ndec = 2, Y = '2.53k'.	*None*