Digital filters

- Low pass (e.g., antialiasing)
- High pass
- Band pass
- Notch
Filter characteristics

- Low pass
- High pass
- Band pass
- Notch

Practical filter characteristic
**Constant bandwidth**

- Centre frequency is a mean of lower and upper cut-off frequency, e.g. FFT

\[
f_c = 0.5(f_l + f_u)
\]

\[
\Delta f = f_u - f_l
\]

\[
f_u = f_l + \Delta f
\]

**Constant % bandwidth**

- Centre frequency is a geometrical mean of lower and upper cut-off frequency

\[
f_c = \sqrt{f_l f_u}
\]

\[
\log f_c = 0.5(\log f_l + \log f_u)
\]

\[
\%BW = \frac{f_u}{f_l}
\]

\[
f_u = f_l \%BW
\]

---

**Constant bandwidth and % bandwidth filters**

- Constant bandwidth (e.g. FFT)

- Constant percentage bandwidth

---

Fig. 3.4: Difference between a constant bandwidth analyzer and a constant percentage bandwidth analyzer [R. Randall, Frequency Analysis &K]
Octave filter

\[
\frac{f_u}{f_i} = 2 \\
\frac{f_c}{f_i} = \sqrt{f_i f_u} = \sqrt{2} f_i \\
\frac{f_i}{\sqrt{2}} \\
f_u = \sqrt{2} f_c
\]

- Centre frequencies are internationally standardised and based on a reference of 1000 Hz

Decade filter

\[
\frac{f_u}{f_i} = 10 \\
\frac{f_c}{f_i} = \sqrt{f_i f_u} = \sqrt{10} f_i \\
\frac{f_i}{\sqrt{10}} \\
f_u = \sqrt{10} f_c
\]

- Also based on the reference of 1000 Hz
General form of a digital filter (see Matlab help on filter)

\[ y(n) = b(1) \cdot x(n) + b(2) \cdot x(n-1) + \ldots + b(nb+1) \cdot x(n-nb) \]
\[ - a(2) \cdot y(n-1) - \ldots - a(na+1) \cdot y(n-na) \]

- where:
  - \( x(n), x(n-1), x(n-2) \) etc. - current and previous inputs to the filter
  - \( y(n), y(n-1), y(n-2) \) etc. - current and previous outputs from the filter
  - **b** coefficients (the so called filter numerator coefficients)
  - **a** coefficients (the so called filter denominator coefficients)

Algorithm

The `filter` function is implemented as a direct form II transposed structure.

The input/output description of this filtering operation is in the z-transform domain as a rational transfer function:

\[ Y(z) = \frac{b(1) + b(2)z^{-1} + \ldots + b(nb+1)z^{-nb}}{1 + a(2)z^{-1} + \ldots + a(na+1)z^{-na}} \cdot X(z) \]

`y = filter(b, a, x)` filters the data in vector `x` with the filter described by numerator coefficient vector `b` and denominator coefficient vector `a`. If \( a(1) \) is not equal to 1, `filter` normalizes the filter coefficients by `a(1)`. If `a(1)` equals 0, `filter` returns an error.
Application of a digital filter

- Example
- A signal in the vector \( g \) was sampled at 44,100 Hz. Apply a third order Butterworth low pass filter with a cut-off frequency of 5,000 Hz

\[
[b, a] = \text{butter}(3, 5000 / (44100 / 2))
\]
\[
g\text{Filtered} = \text{filter}(b, a, g)
\]

IIR and FIR filters

FIR filter (finite impulse response)
- when \( a(1) = 1 \) and other \( a \) coefficients are = 0
- otherwise

IIF filter (infinite impulse response)

Moving average as a digital filter

Example
You can use \( \text{filter} \) to find a running average without using a \( for \) loop. This example finds the running average of a 16-element vector, using a window size of 5.

\[
data = [1:0.2:4]';
windowSize = 5;
filter(ones(1,windowSize)/windowSize,1,data)
\]