Signal Processing

Please send corrections (or suggestions) to youngwlim@hotmail.com.

Based on

Signal Processing with Free Software : Practical Experiments
F. Auger

filter (1)

```
: y = filter (b, a, x)
: [y, sf] = filter (b, a, x, si)
: [y, sf] = filter (b, a, x, [], dim)
: [y, sf] = filter (b, a, x, si, dim)
```

https://octave.sourceforge.io/octave/function/filter.html

filter (2)

Apply a 1-D digital filter to the data x.

filter returns the solution to the following
linear, time-invariant difference equation:

$$\sum_{k=0}^{N} a(k+1)\, y(n-k) \;=\; \sum_{k=0}^{M} b(k+1)\, x(n-k) \qquad \mathit{for}\;\; 1 \le n \le \mathit{length}(x)$$

where N=length(a)-1 and M=length(b)-1.

$$\boldsymbol{a} = [a(1), a(2), \cdots, a(N+1)]$$
$$\boldsymbol{b} = [b(1), b(2), \cdots, b(M+1)]$$
$$\mathit{length}(\boldsymbol{a}) = N+1$$
$$\mathit{length}(\boldsymbol{b}) = M+1$$

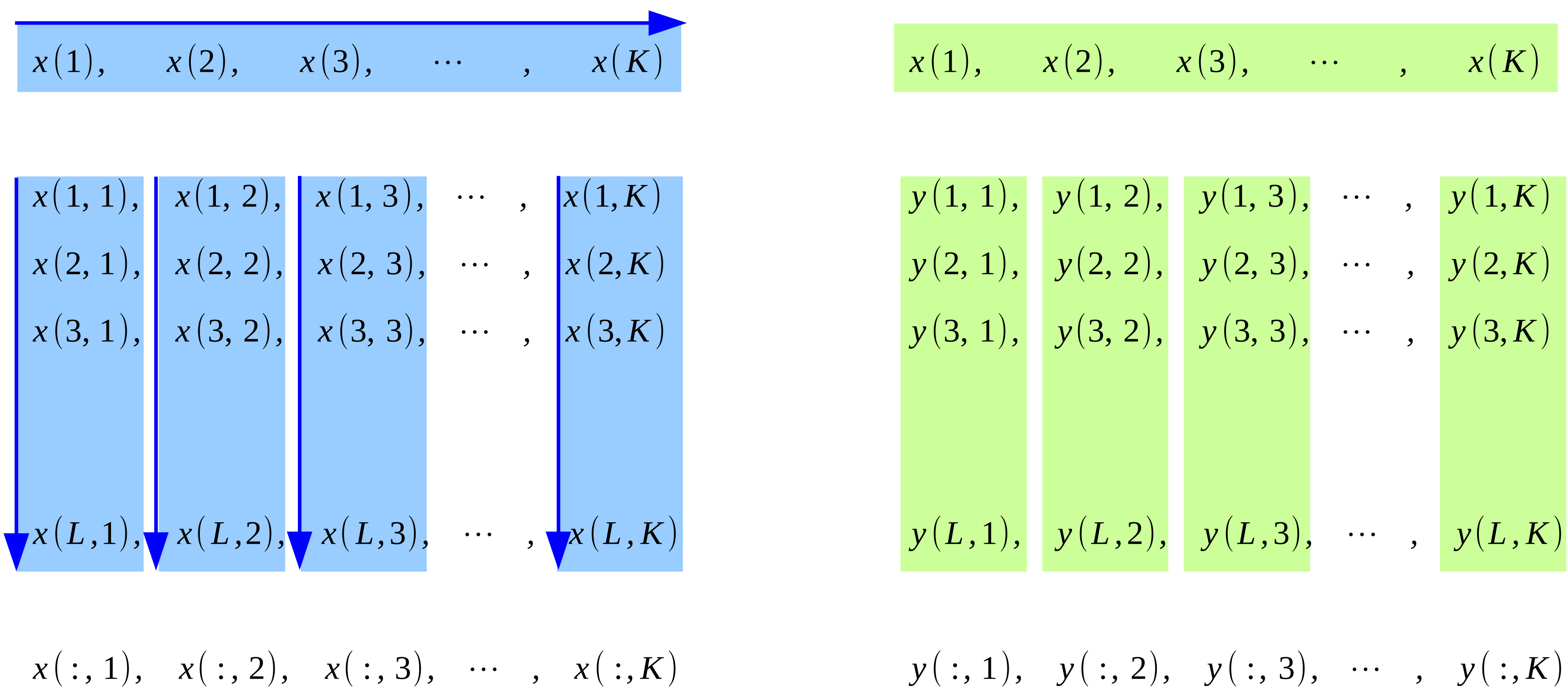
$$\boldsymbol{x} = [x(1), x(2), \cdots, x(L+1)]$$
$$\mathit{length}(\boldsymbol{x}) = L+1$$

$$1 \le n \le L+1$$

https://octave.sourceforge.io/octave/function/filter.html

filter (3)

The result is calculated over the **first** non-singleton dimension of x
or over **dim** if supplied.



https://octave.sourceforge.io/octave/function/filter.html

filter (4)

$$\sum_{k=0}^{N} a(k+1)\, y(n-k) = \sum_{k=0}^{M} b(k+1)\, x(n-k) \qquad for \;\; 1 \le n \le length(x)$$

$$a(1)\, y(n) + \sum_{k=1}^{N} a(k+1)\, y(n-k) = \sum_{k=0}^{M} b(k+1)\, x(n-k)$$

$$a(1)\, y(n) = -\sum_{k=1}^{N} a(k+1)\, y(n-k) + \sum_{k=0}^{M} b(k+1)\, x(n-k)$$

$$y(n) = -\sum_{k=1}^{N} \frac{a(k+1)}{a(1)}\, y(n-k) + \sum_{k=0}^{M} \frac{b(k+1)}{a(1)}\, x(n-k)$$

$$y(n) = -\sum_{k=1}^{N} c(k+1)\, y(n-k) + \sum_{k=0}^{M} d(k+1)\, x(n-k) \qquad for\, 1 \le n \le length(x)$$

where c = a/a(1) and d = b/a(1).

https://octave.sourceforge.io/octave/function/filter.html

filter (5)

si : the initial state of the system
sf : the final state
The state vector is a column vector
whose length is equal to the length of
the longest coefficient vector minus one.

No **si** is presented, the zero initial state.

In terms of the Z Transform,
y is the result of passing the discrete-time signal **x**
through a system characterized
by the following rational system function:

$$H(z) = \frac{\sum_{k=0}^{M} d(k+1) z^{-k}}{1 + \sum_{k=1}^{N} c(k+1) z^{-k}}$$

https://octave.sourceforge.io/octave/function/filter.html

freqz (1)

```
: [h, w] = freqz (b, a, n, "whole")
: [h, w] = freqz (b)
: [h, w] = freqz (b, a)
: [h, w] = freqz (b, a, n)
: h = freqz (b, a, w)
: [h, w] = freqz (..., Fs)
: freqz (...)
```

https://octave.sourceforge.io/octave/function/freqz.html

freqz (2)

Return the complex frequency response h
of the rational **IIR** filter
with the numerator coefficients **b** and
the denominator coefficients **a**

The response is evaluated
at **n** angular frequencies between **0** and **2*pi**.

The output value **w** is a vector of the frequencies.

https://octave.sourceforge.io/octave/function/freqz.html

freqz (3)

If **a** is omitted, the denominator is assumed to be **1**
(this corresponds to a simple **FIR** filter).

If **n** is omitted, a value of **512** is assumed.
For fastest computation, **n** should factor into a small number of small primes.

If the fourth argument, "**whole**", is omitted
the response is evaluated at frequencies between **0** and **pi**.

https://octave.sourceforge.io/octave/function/freqz.html

freqz (4)

freqz (b, a, w)

Evaluate the response at the specific frequencies in the vector **w**.
The values for **w** are measured in radians.

[…] = **freqz** (…, Fs)

Return frequencies in Hz instead of radians assuming a sampling rate Fs.
If you are evaluating the response at specific frequencies **w**,
those frequencies should be requested in Hz rather than radians.

freqz (…)

Plot the magnitude and phase response of h rather than returning them.

https://octave.sourceforge.io/octave/function/freqz.html

freqz_plot

: **freqz_plot** (w, h)
: **freqz_plot** (w, h, freq_norm)

Plot the magnitude and phase response of h.

If the optional freq_norm argument is true,
the frequency vector w is in units of normalized radians.
If freq_norm is false, or not given, then w is measured in Hertz.

https://octave.sourceforge.io/octave/function/freqz_plot.html

conv

```
: conv (a, b)
: conv (a, b, shape)
```

Convolve two vectors a and b.

The output convolution is a vector with length equal to
length (a) + length (b) - 1.
When a and b are the coefficient vectors of two polynomials,
the convolution represents the coefficient vector of the product polynomial.

The optional shape argument may be

shape = "full"

Return the full convolution. (default)

shape = "same"

Return the central part of the convolution with the same size as a.

https://octave.sourceforge.io/octave/function/conv.html

fftconv

: **fftconv** (x, y)
: **fftconv** (x, y, n)

Convolve two vectors using the FFT for computation.

c = fftconv (x, y) returns a vector of length equal to length (x) + length (y) - 1. If x and y are the coefficient vectors of two polynomials, the returned value is the coefficient vector of the product polynomial.

The computation uses the FFT by calling the function fftfilt. If the optional argument n is specified, an N-point FFT is used.

See also: deconv, conv, conv2.

https://octave.sourceforge.io/octave/function/fftconv.html

deconv

: **deconv** (y, a)

Deconvolve two vectors.

[b, r] = deconv (y, a) solves for b and r such that y = conv (a, b) + r.

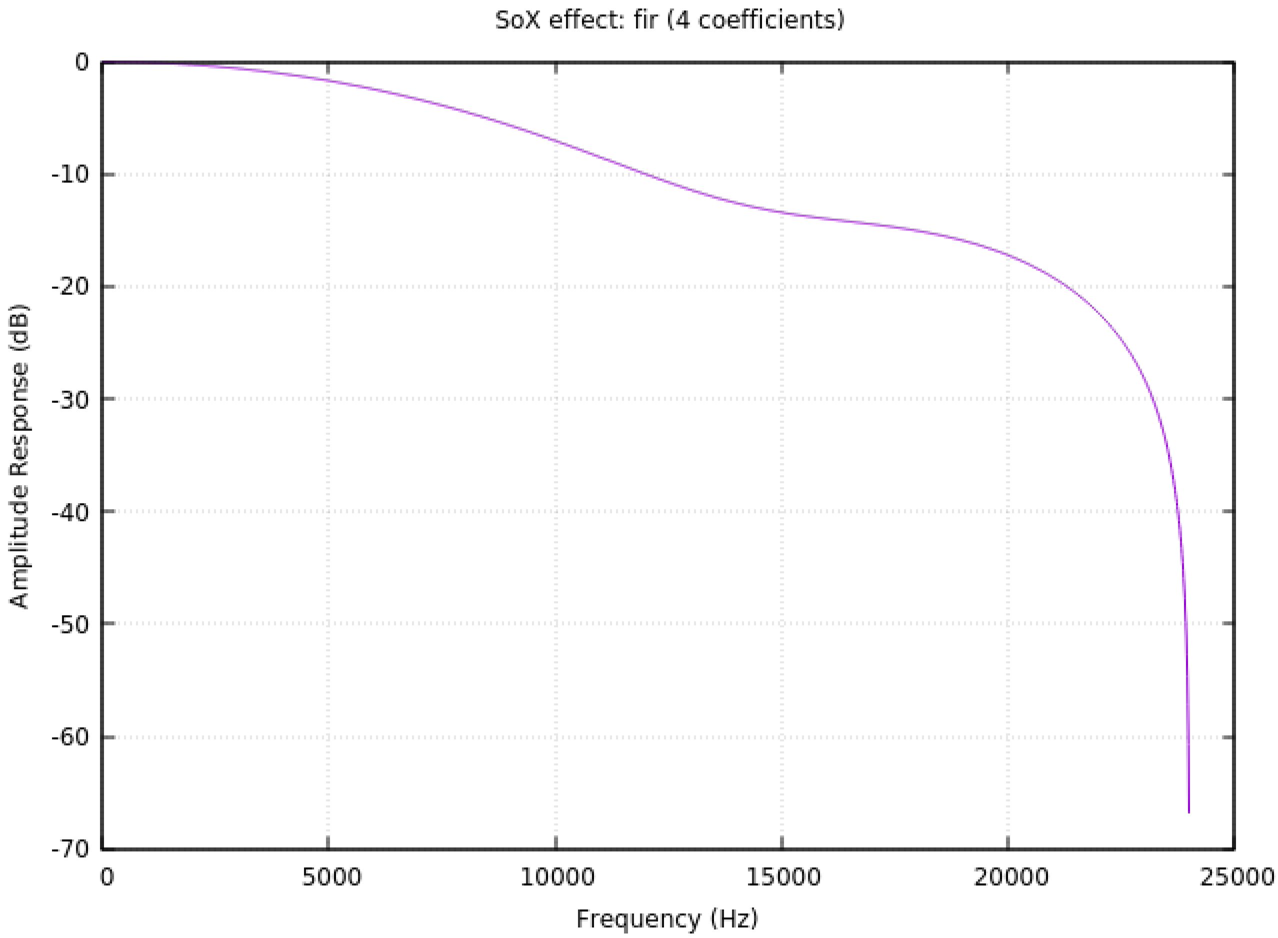
If y and a are polynomial coefficient vectors, b will contain the coefficients of the polynomial quotient and r will be a remainder polynomial of lowest order.

https://octave.sourceforge.io/octave/function/deconv.html

--plot gnuplot | octave

```
sox --plot gnuplot s6s.wav -n fir 0.1 0.2 0.4 0.3          >fir1.plt
sox --plot gnuplot s6s.wav -n fir coeff.txt                >fir2.plt
sox --plot gnuplot s6s.wav -n biquad .6 .2 .4 1 -1.5 .6 >fir3.plt
sox --plot gnuplot s6s.wav -n fir 0.2 0.2 0.2 0.2 0.2    >fir4.plt
```

--plot gnuplot | octave



References

[1] F. Auger, Signal Processing with Free Software : Practical Experiments