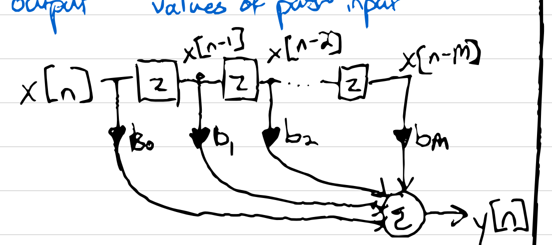
Data Filtering

Two Categories of Discrete-Time Filters:

Finite Impulse Response (FIR)

$y[n] = \sum_{k=0}^{m} b_x x[n-k]$ moving average

$y[n]$: some output; $\sum b_x x[n-k]$: sum of the weighted values of past input



$H(z) = \sum_{k=0}^{M} b_k z^{-k}$

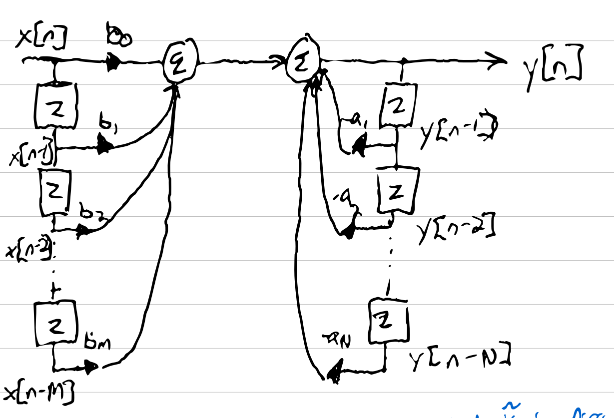
$H(z)$: Freq. Response

Optimized based design to find desired response, $\tilde{H}(z)$
- Arbitrary magnitude/phase response
- Can obtain linear phase shift
- Can require large M to work

Infinite Impulse Response (IIR)

$y[n] = \sum_{k=0}^{M} b_x x[n-k] - \sum_{k=1}^{N} a_k y[n-k]$

past inputs; past outputs

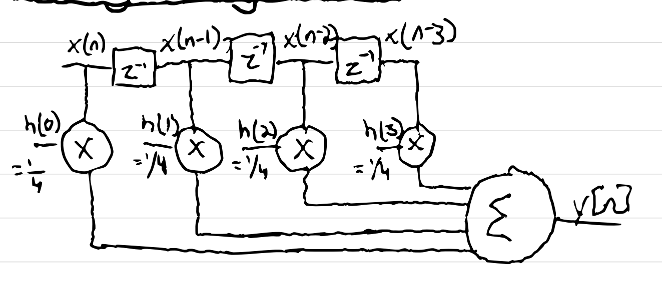


$H(z) = \dfrac{\sum_{k=0}^{M} b_k z^{-k}}{\sum_{k=0}^{N} a_k z^{-k}}, \quad a_0 = 1$

To find $\tilde{H}$ is difficult via optimization since local minima and maxima exist, need to restrict a, b

Transform analog filter design to find $\tilde{H}(z)$
- Frequency selective gain
- Nonlinear phase, no control
- Fewer coefficients than FIR

Moving Average Filter:



$y(n) = \frac{1}{N} \sum_{k=0}^{N-1} x(n-k)$ where N = number of samps to avg.

Example:

$y(0) = NaN$ x_0

$y(1) = NaN$ $x_1 + x_0$

$y(2) = NaN$ $x_2 + x_1 + x_0$

$y(3) = 0.25(x_3 + x_2 + x_1 + x_0)$

$y(4) = 0.25(x_4 + x_3 + x_2 + x_1)$

$y(5) = 0.25(x_5 + x_4 + x_3 + x_2)$

$y(6) = 0.25(x_6 + x_5 + x_4 + x_3)$

$y(7) = 0.25(x_7 + x_6 + x_5 + x_4)$

- Useful for eliminiating noise, but preserving pulses.
- Rise/Fall time equals $N \cdot t_s$
- Frequency response of filter has a sharper dropoff w/ increasing N, but has "noisy" side lopes that hinder stopband attenuation (i.e. not ideal)

Pseudo Code

Real-Time

```
for k < N,
    obtain sample
    save to buffer
return average of buffer
```

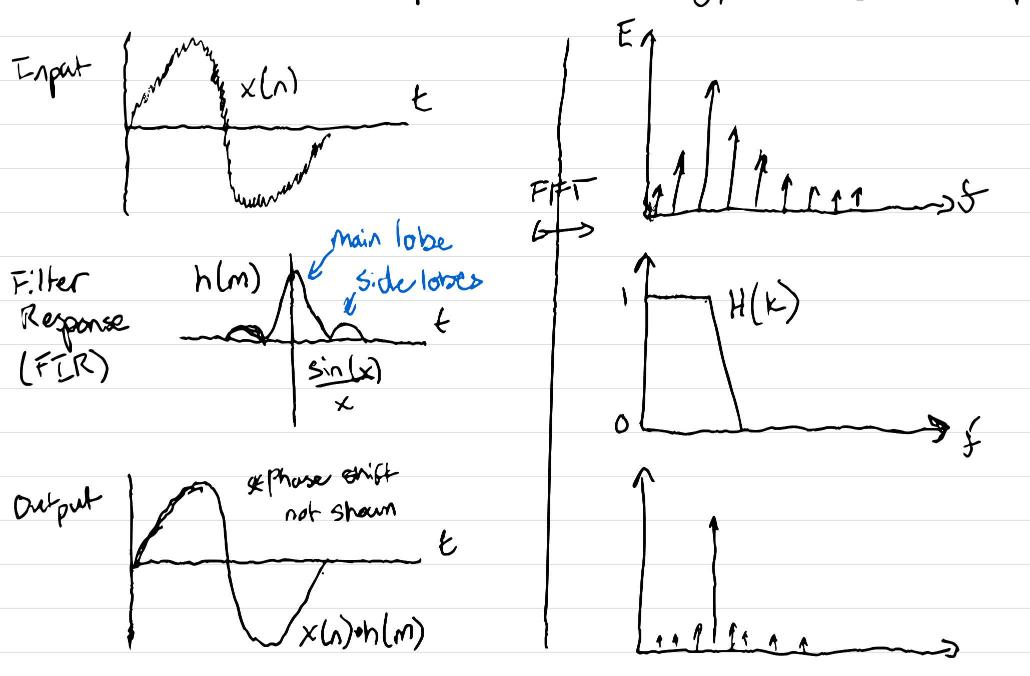
Post-Processing

```
load sample vector from memory
for i=0:M-N
    for k<N
        select x_k
        append to buffer
    take average of buffer
    append to output buffer
```

Basic Terminology

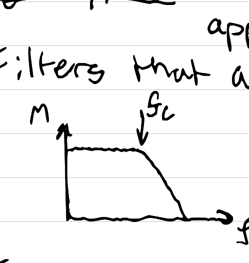
Time-domain: all measurements are made in terms of time and magnitude e.g. accelerations, wave heights, etc.

Frequency Domain: all measurements are made in terms of frequency (or period) and energy e.g. wave spectra.

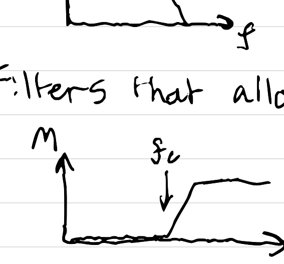


Cutoff Frequency, f_c: The frequency at which the filter stops applying

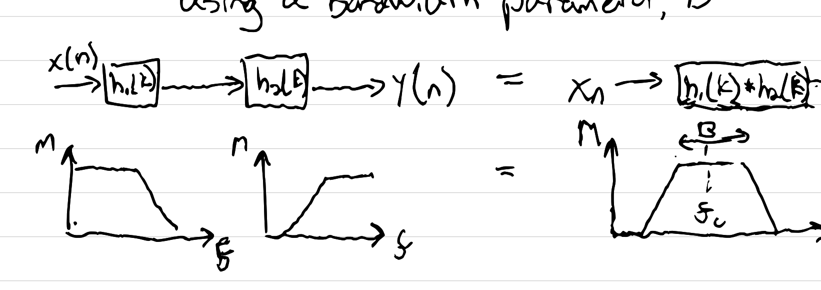
Lowpass: Filters that all signals that are lower than f_c



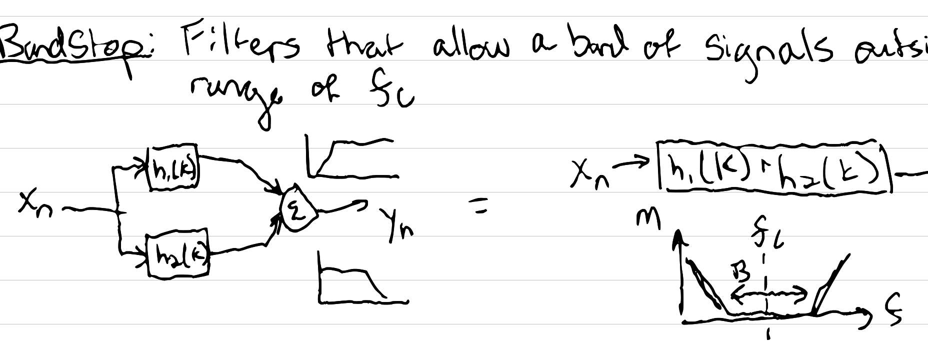
Highpass: Filters that allow signals that are heigher than f_c



Band Pass: Filters that allow a band of signals around f_c using a bandwidth parameter, B



BandStop: Filters that allow a band of signals outside the range of f_c

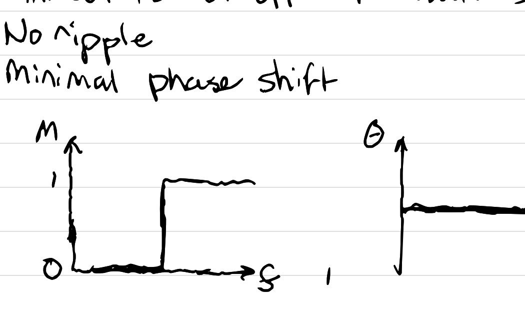


Frequency Response, H: How signal frequencies react to the filter

Phase Shift: How the time series is shifted by the filter.

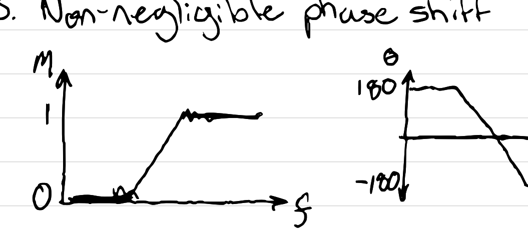
Ideal Filter:
1. Immediate roll-off (attenuation)
2. No ripple
3. Minimal phase shift

Bode Plot:

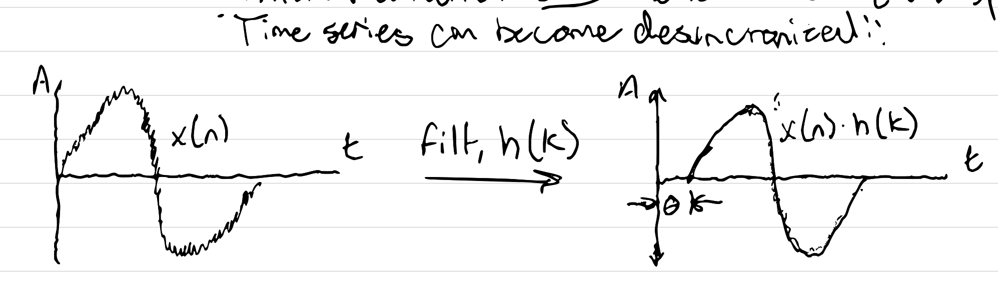


Realistic Filter:
1. Delayed roll-off
2. Ripple around f_c, stop bands
3. Non-negligible phase shift

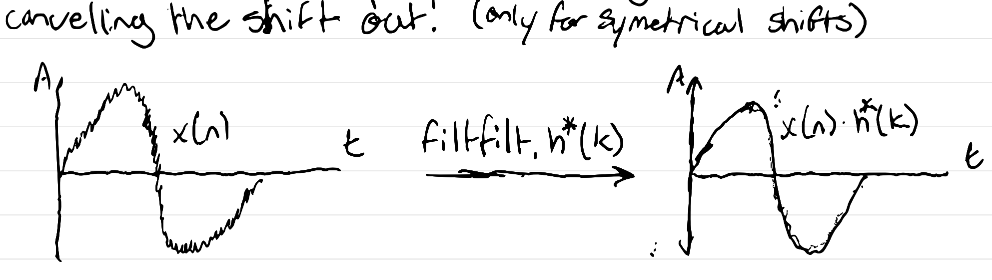
Bode Plot:



Implications:
- Some signals unintentionally over/under attenuated
- Intensification of some noise around f_c and stop bands
- Time series can become desyncronized!

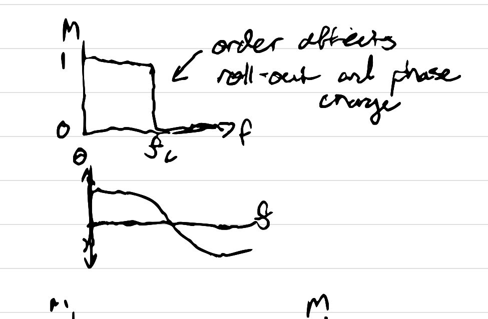


How can improve the phase difference? By applying the filtering technique forwards and backwards! This applies the phase shift in both positive and negative directions, effectively cancelling the shift out! (only for symetrical shifts)

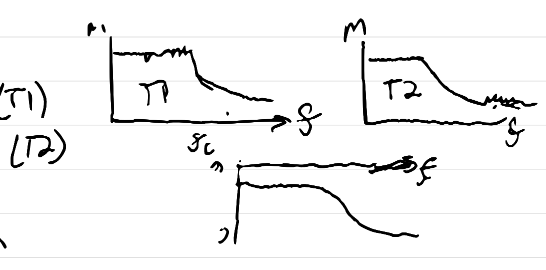


IIR Filters:

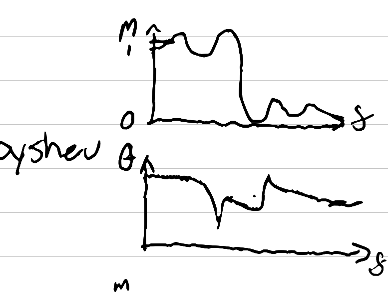
Butterworth:
- Simple, but effective
- No ripples in passband or stopband
- Maximally flat response
- Maximum phase shift



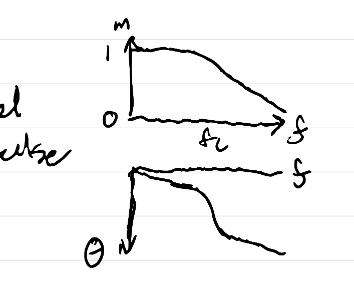
Chebyshev:
- Ripple in the passband, not stopband (T1)
- Ripple in the stopband, not passband (T2)
- Less phase shift than Butterworth
- Less responsive than Butterworth



Elliptical:
- Ripples in stopband and passband
- Shorter transition region than Chebyshev
- Degraded Phase response



Bessel:
- No ripples in passband or stopband
- Optimized for linear phase and pulse response
- Longest transition region



In MATLAB:

```
Clear all; close all; clf;

fc = 50
fs = 6283 (sample @ 2000π Hz)

[b,a] = butter(6, fc/(fs/2)); % Butterworth of order 6

freqz(b,a,[],fs) % Bode Plot
subplot(2,1,1)
ylim([-100, 20])

x = 0:0.001:2π
data_in = sin(10·x) + randn(size(x))·0.1
data_out = filtfilt(b,a,data_in);

figure
hold on
plot(x, data_in)
plot(x, data_out)
plot(x, sin(10·x))
legend("raw", "filtered", "true")
xlabel("Time [s]")
ylabel("Surface Elevation [m]")
hold off
```