Eng. Julian S. Bruno

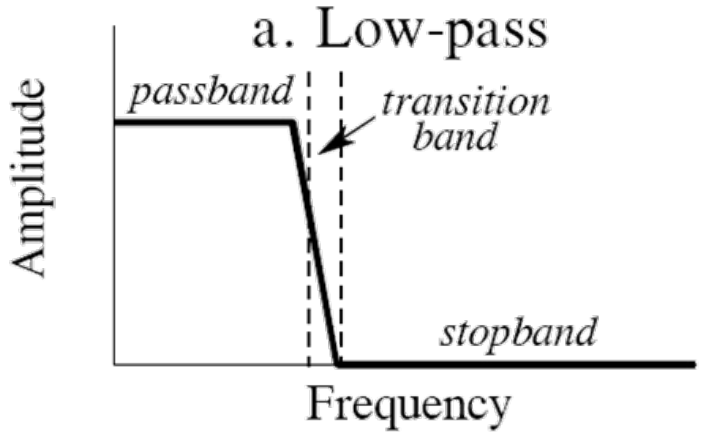
REAL TIME DIGITAL SIGNAL PROCESSING

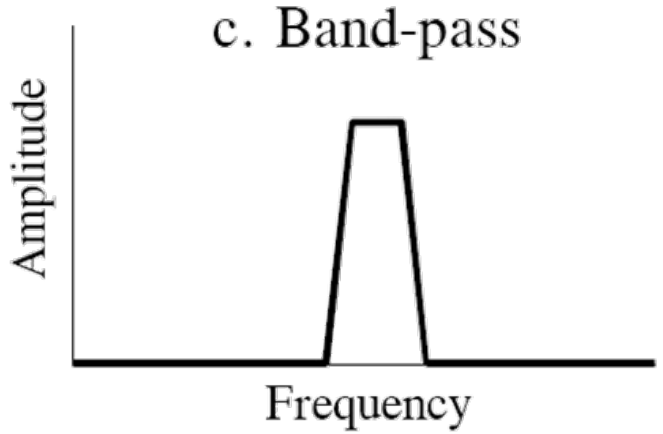
Digital Filters

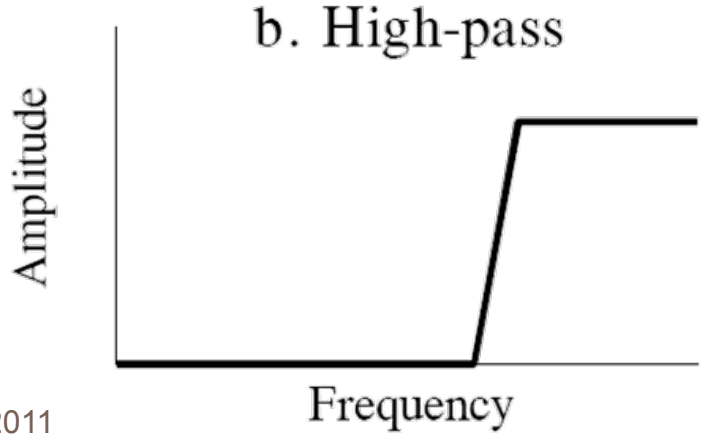
FIR and IIR.

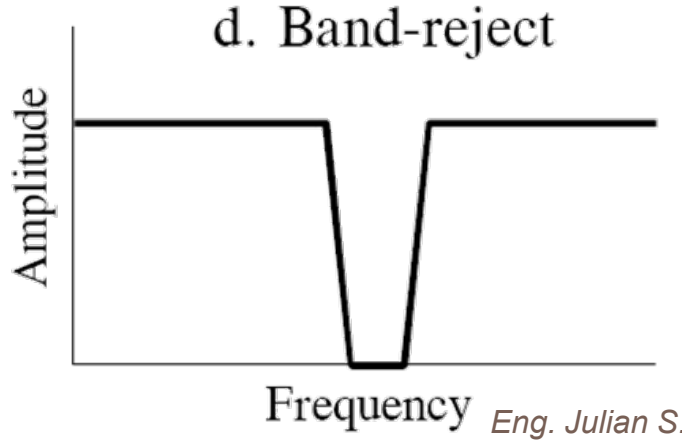
- Design parameters.
- Implementation types.
- Constraints.

Filters: General classification









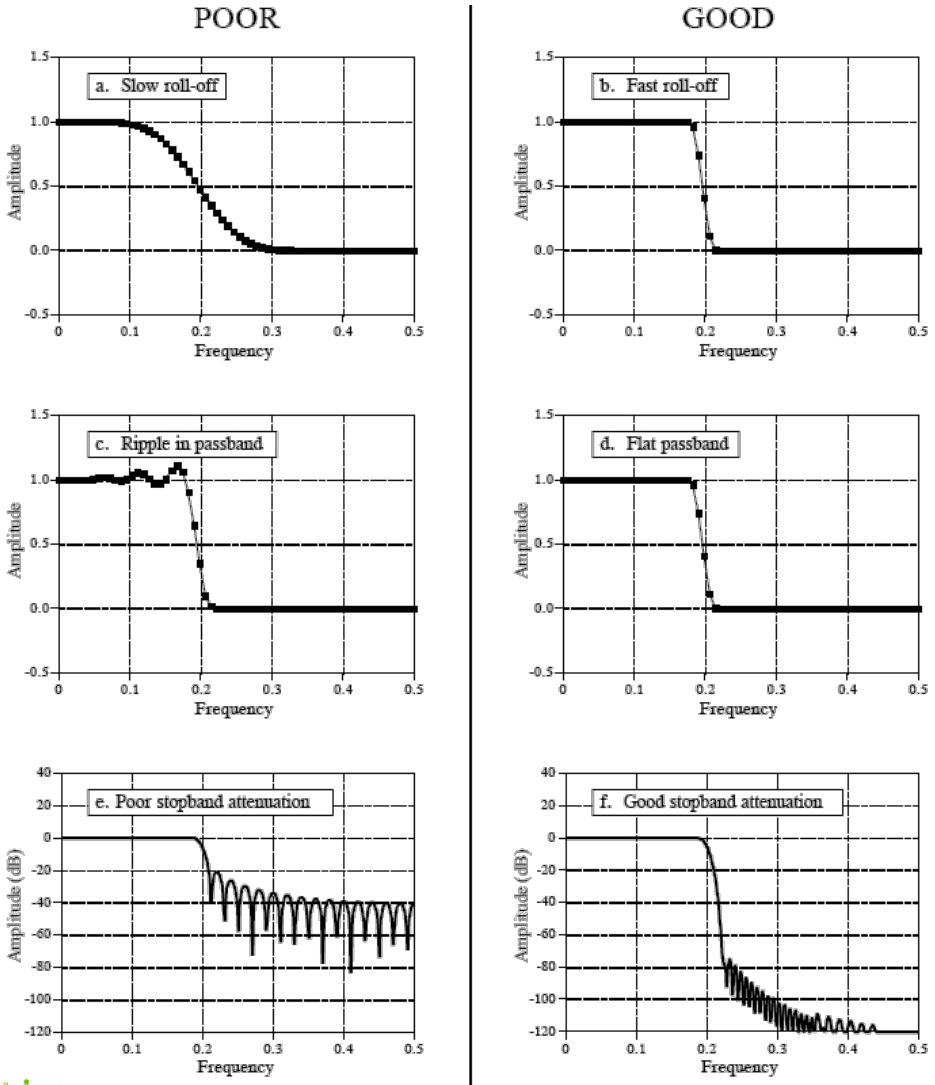
Filters: General classification

FILTER USED FOR:	FILTER IMPLEMENTED BY: Convolution *Finite Impulse Response (FIR)*	Recursion *Infinite Impulse Response (IIR)*
Time Domain *(smoothing, DC removal)*	Moving average (Ch. 15)	Single pole (Ch. 19)
Frequency Domain *(separating frequencies)*	Windowed-sinc (Ch. 16)	Chebyshev (Ch. 20)
Custom *(Deconvolution)*	FIR custom (Ch. 17)	Iterative design (Ch. 26)

SW Smith, The Scientist and Engineer's guide to DSP. California Tech. Pub. 1997.

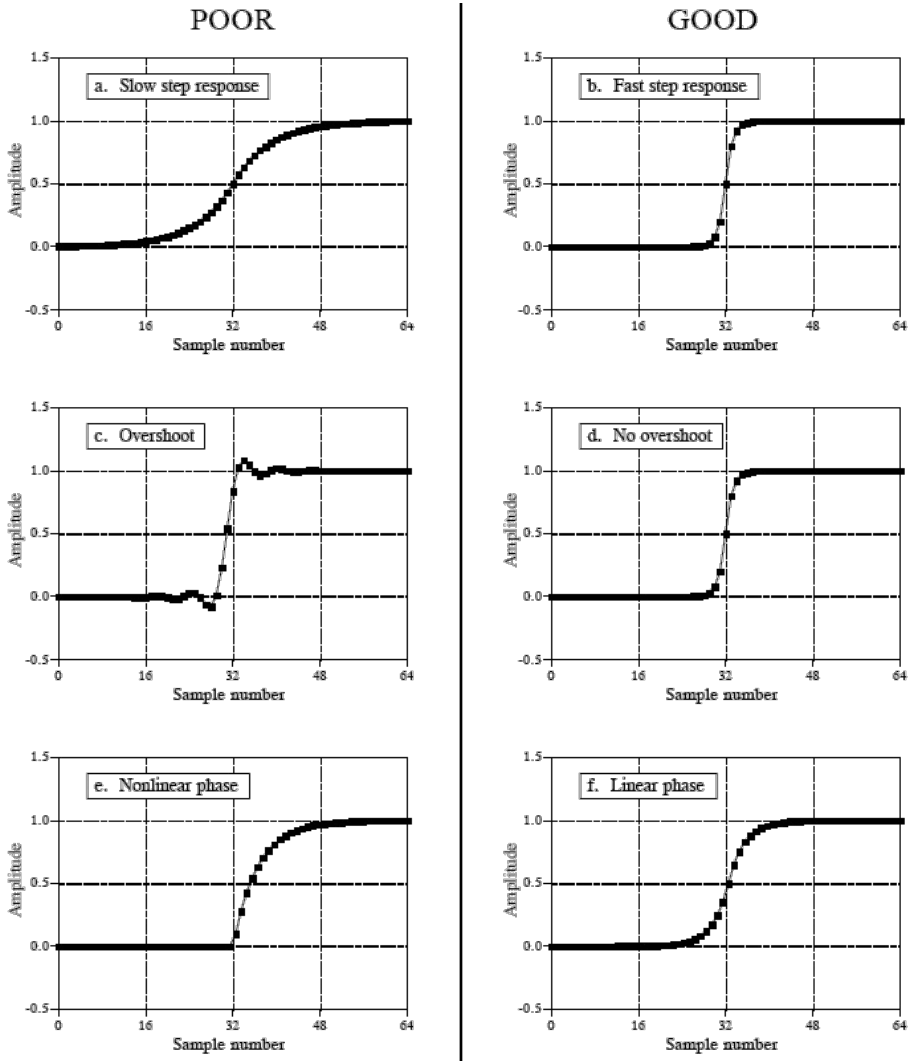
Filters: Frequency domain features

- Fast roll-off, flat passband and high attenuation in stopband are desirable features.
- They can't be achieved at the same time, so the design procedure is a ***trade off*** between these features.



Filters: Time domain features

- Step response evidence the most important time domain features.
- In this case the trade off is between fast step response and overshoot.
- Overshoot is an undesirable distortion and must be avoided.
- Simetry is a “fingerprint” of a linear phase response system (FIR).



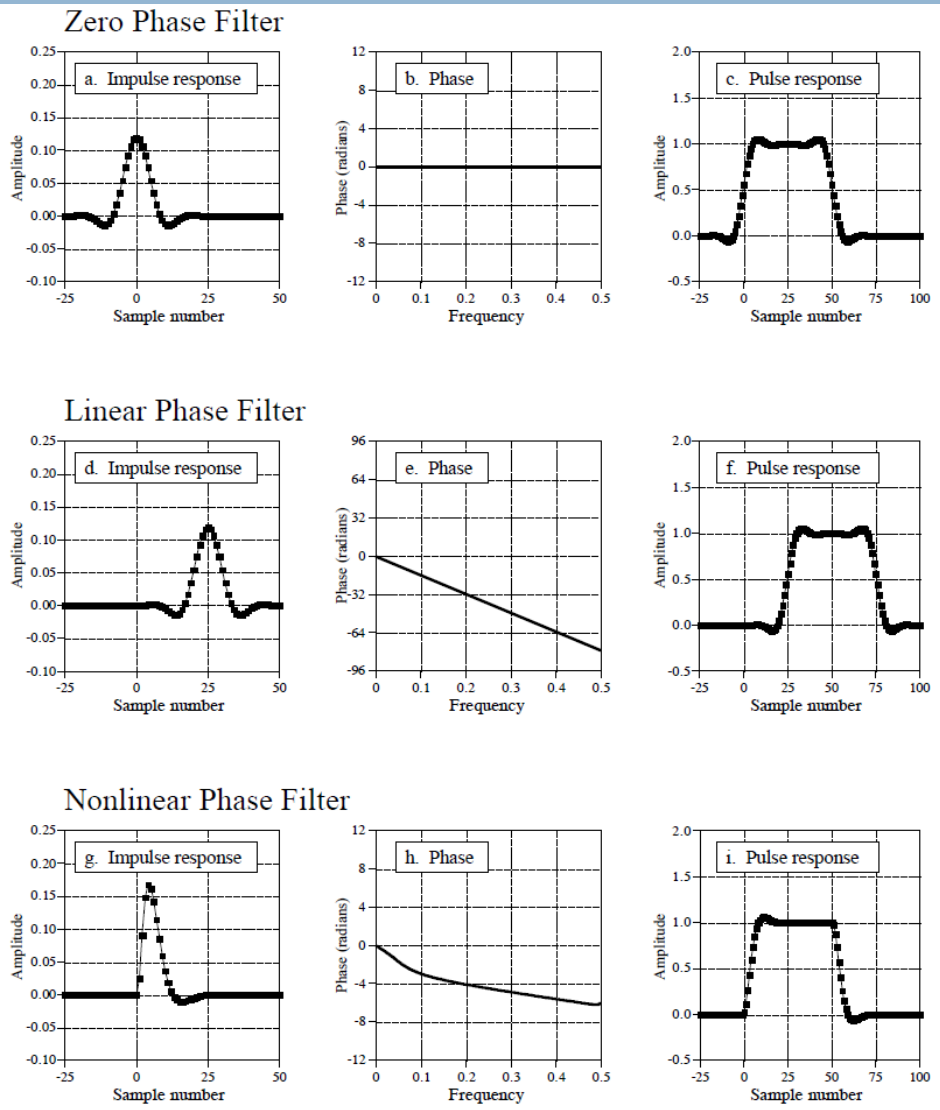
Filters: Phase response

Phase Delay:

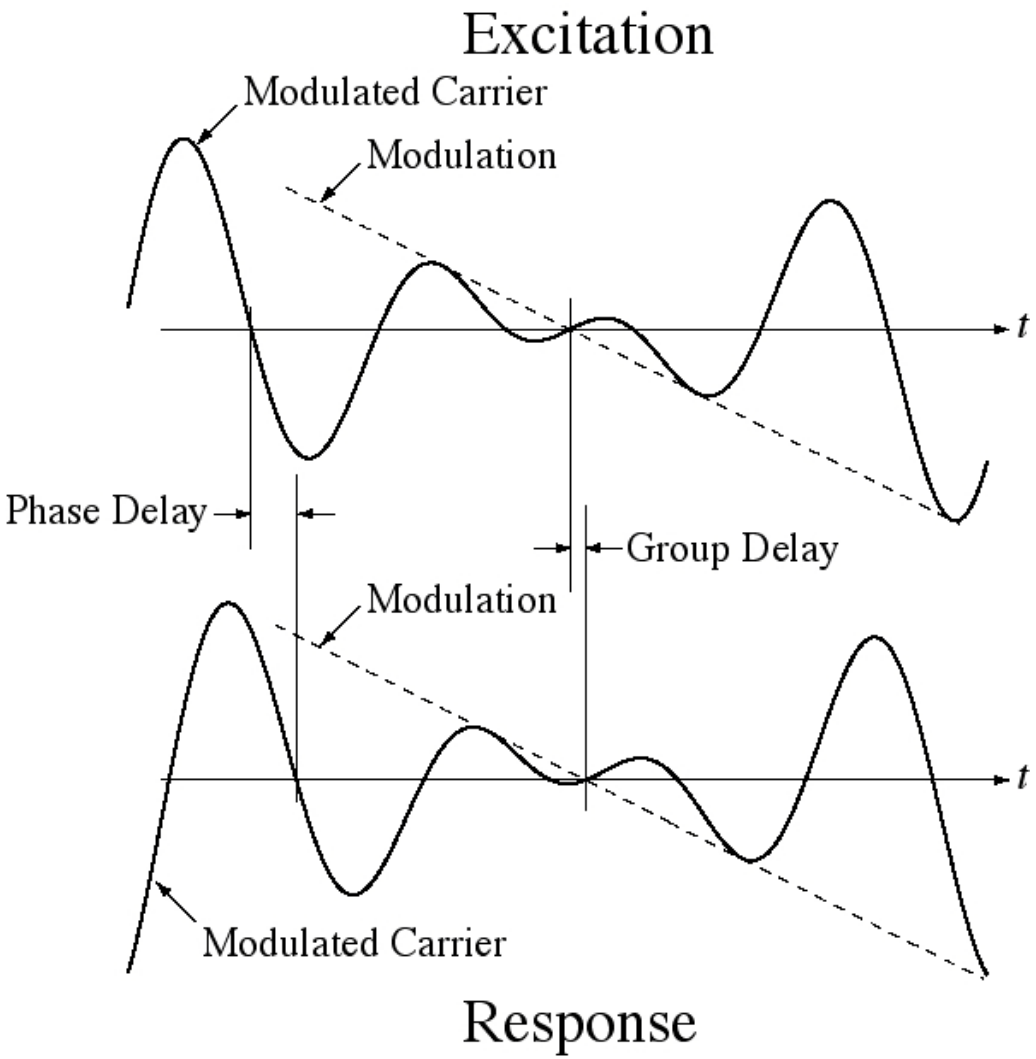
$$pd = -\frac{\varphi_{(\omega)}}{\omega}$$

Group Delay:

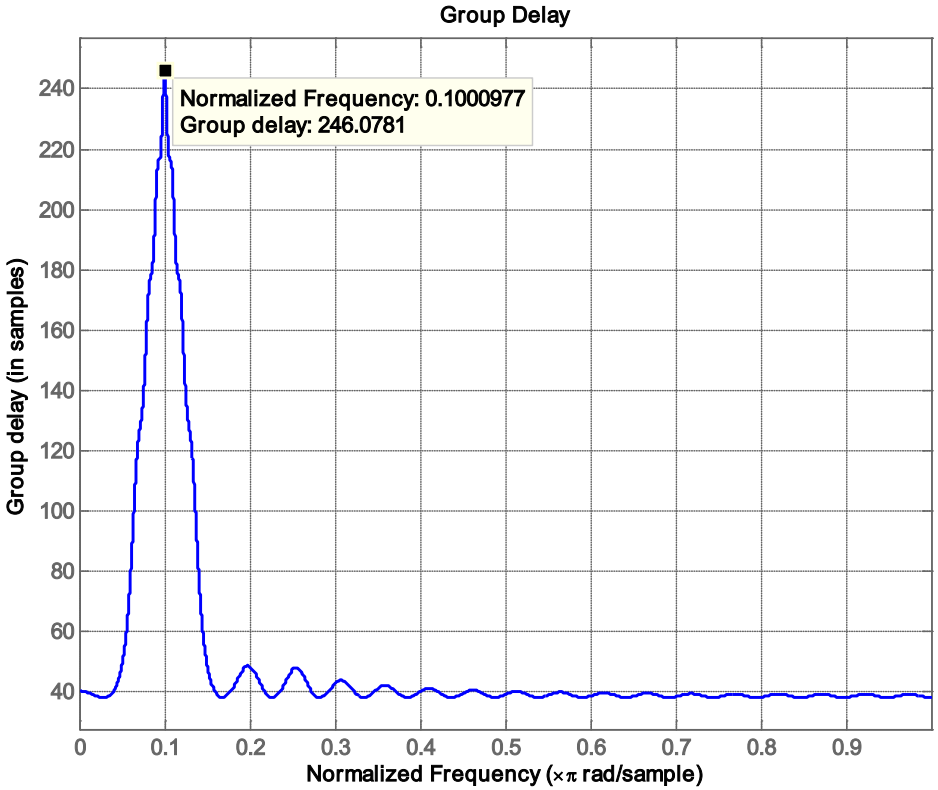
$$gd = -\frac{\partial\varphi_{(\omega)}}{\partial\omega}$$



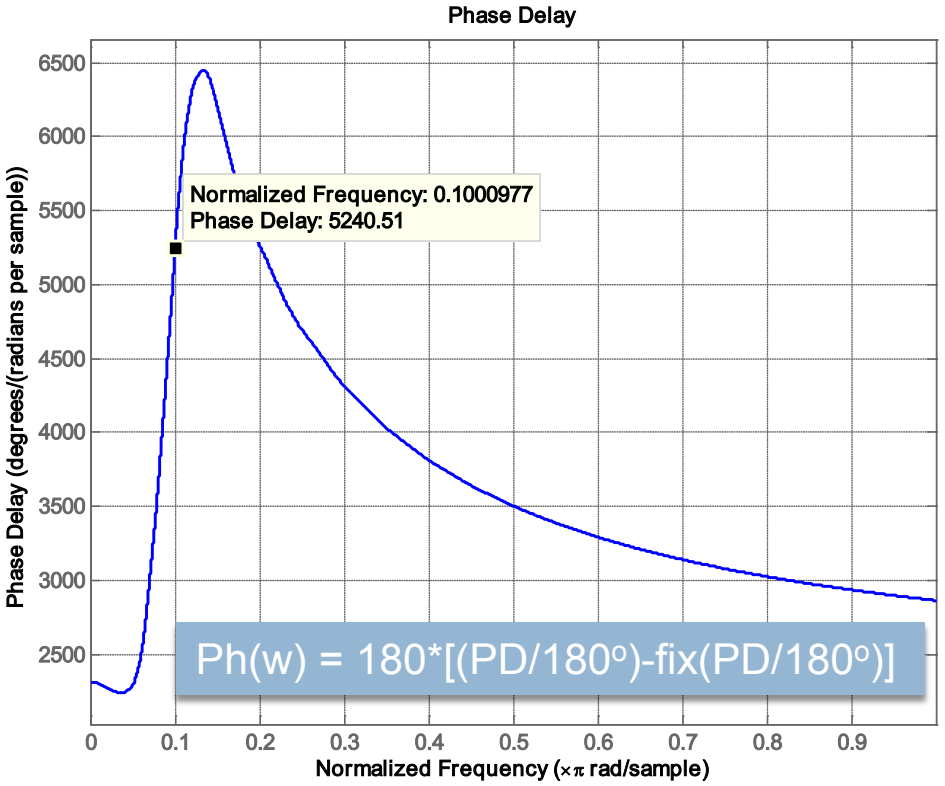
Filters: Group and Phase Delay



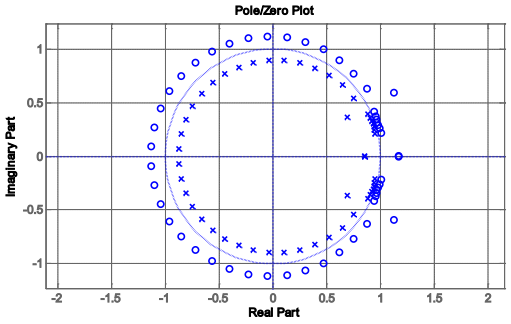
Filters: Group and Phase Delay



|H(w)| = 1

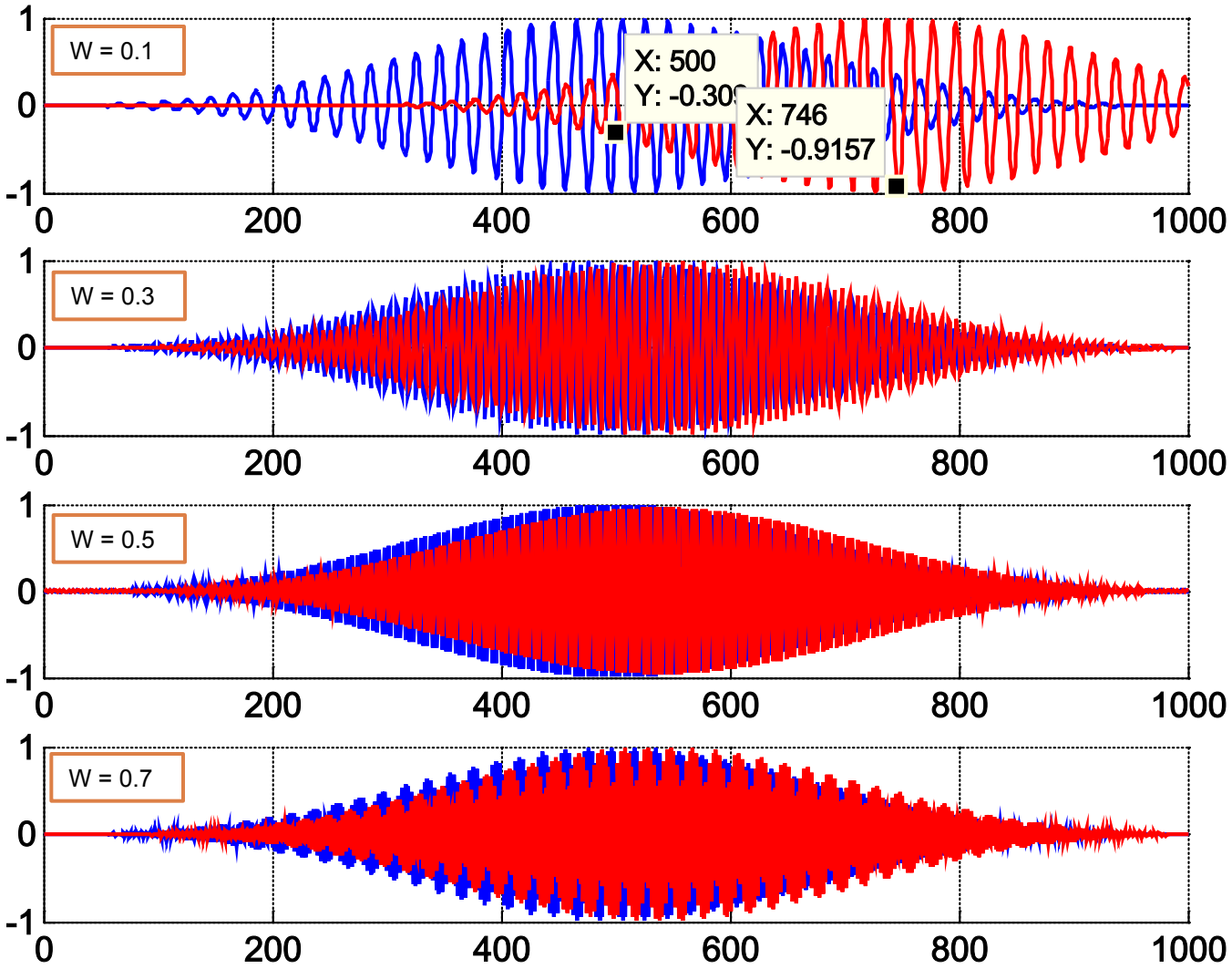


Ph(w) = 180*[(PD/180º)-fix(PD/180º)]



rtdsp

Filters: Group and Phase Delay



Gd = 746-500
Gd = 246

(5240.51/180) = 29,11394

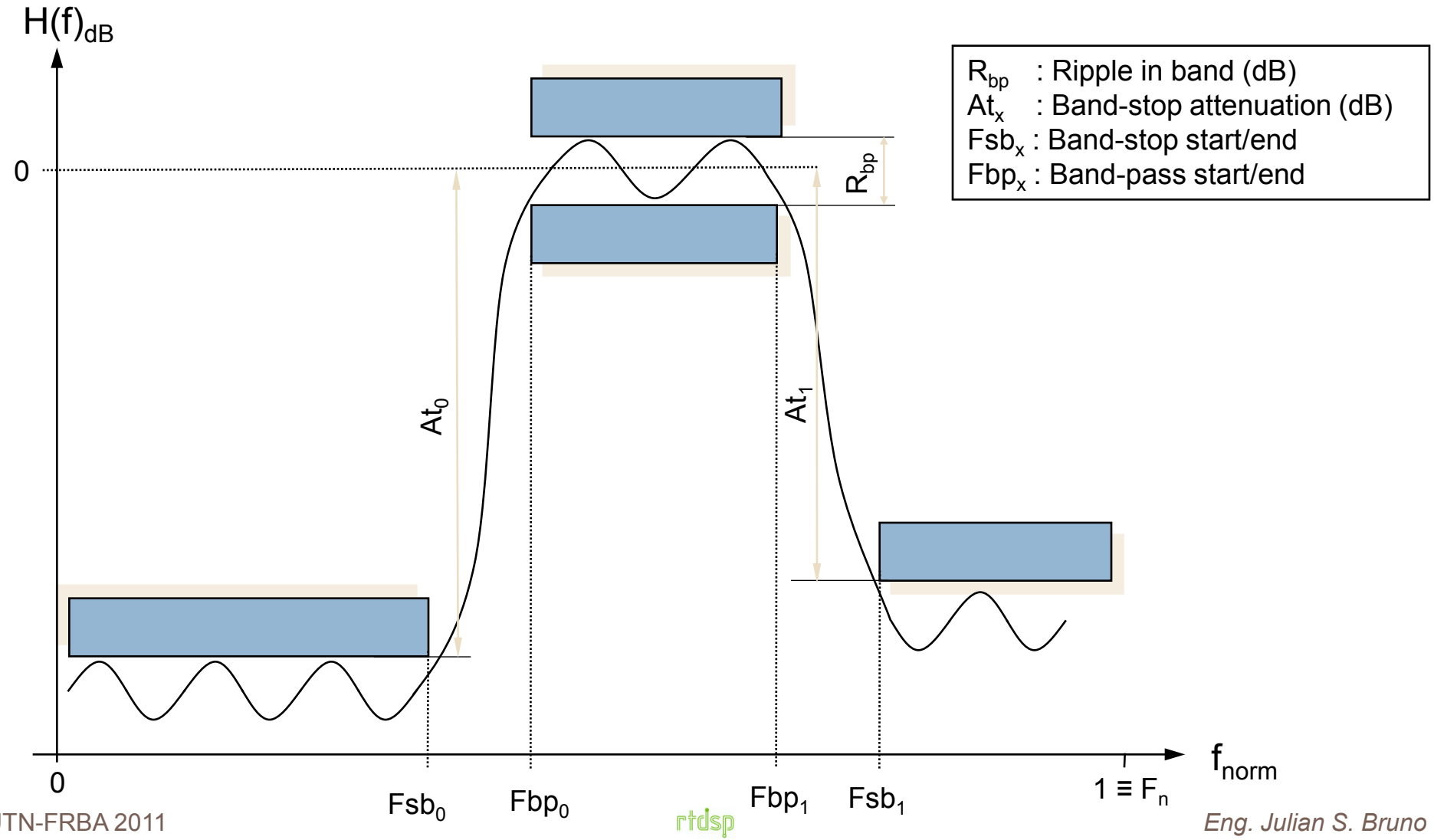
Ph = 0.11394*180
Ph = 20.51

Digital Filters: Design steps

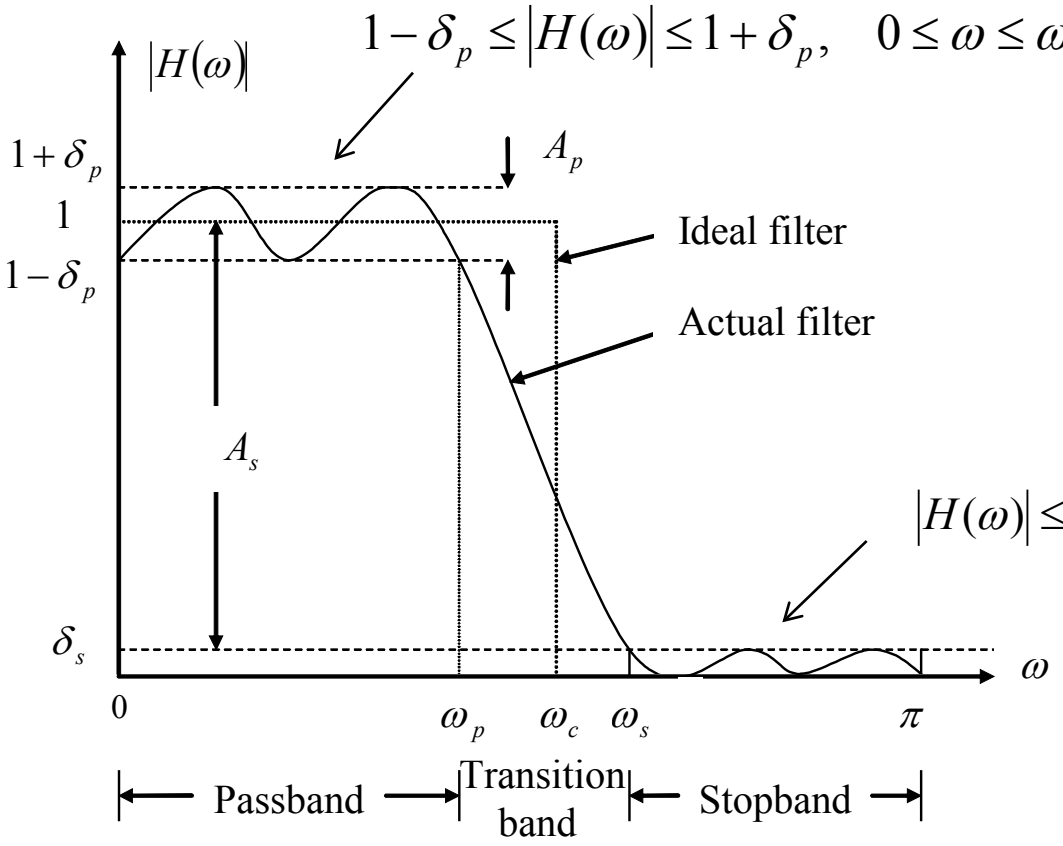
- Requirement specifications.
- Coefficients calculation.
- Filter realization:
  - Structure.
  - Wordlength effects.
- Software or Hardware implementation.

EC Ifeachor, BW Jervis. *Digital Signal Processing. A Practical approach*. Second Edition. Prentice Hall.

Digital Filters: Specifications



Digital Filters: Specifications



$$A_p = 20\log_{10}\left(\frac{1+\delta_p}{1-\delta_p}\right) \text{ dB}$$

$$A_s = -20\log_{10}\delta_s \text{ dB}$$

Finite Impulse Response

FIR

Eng. Julian S. Bruno

FIR filters: General characteristics

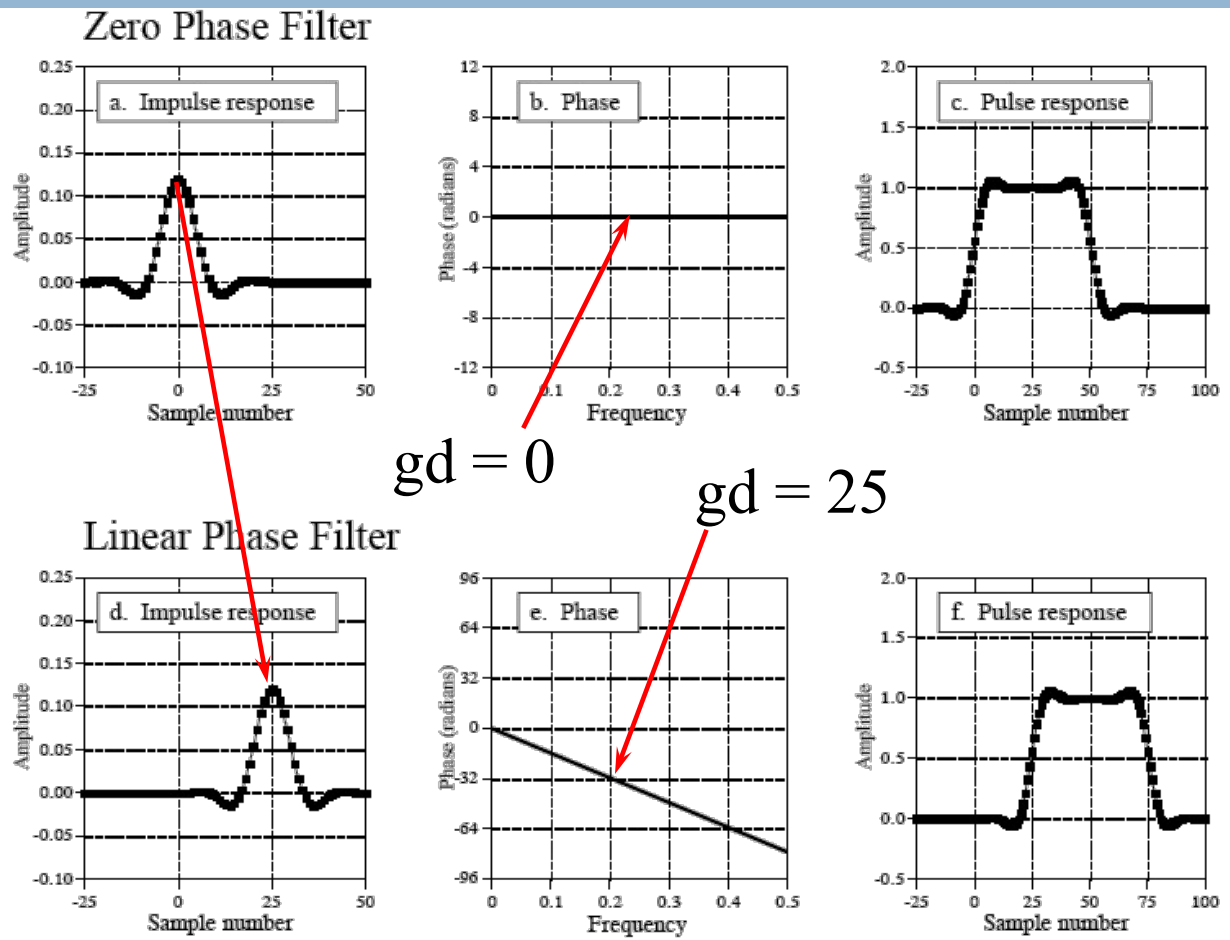
- FIR digital filters use only ***current*** and ***past input samples***. Require no feedback.
- Their transfer function has ***zeros around the Z plane*** and ***poles at the origin***, so FIR filters are ***always stable***.
- They can easily be designed to be ***linear phase and constant delay*** by making the coefficient sequence symmetric.
- Finite duration of nonzero input values / Finite duration of nonzero output values. ***Finite Impulse Response.***

FIR filters: Perfect linear phase

$$pd = gd = -\frac{\partial \varphi_{(f)}}{\partial f}$$

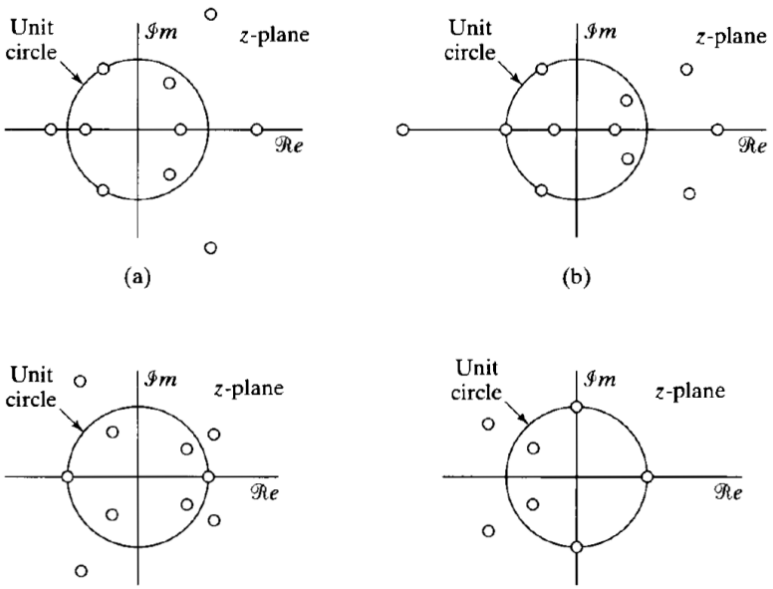
$$delay = \frac{M-1}{2}[samples]$$

- Zero phase filters are a particular case of linear phase with zero delay.

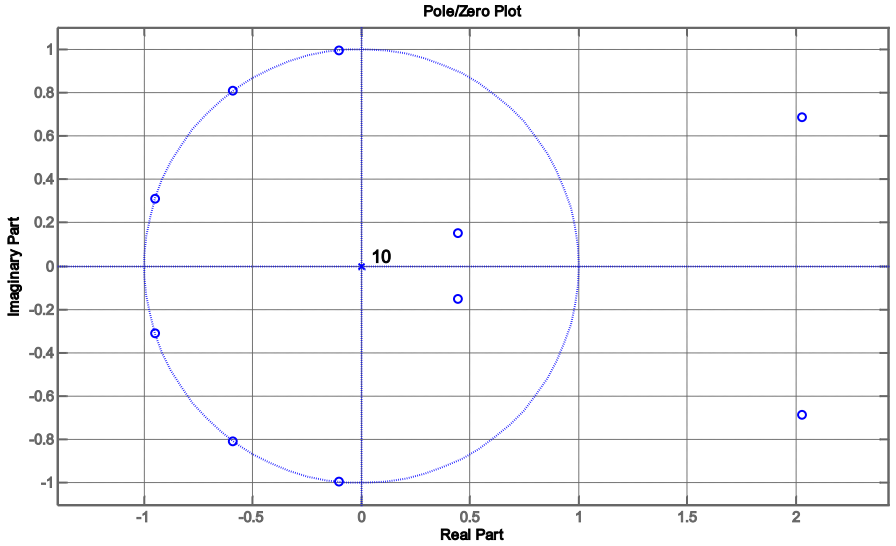


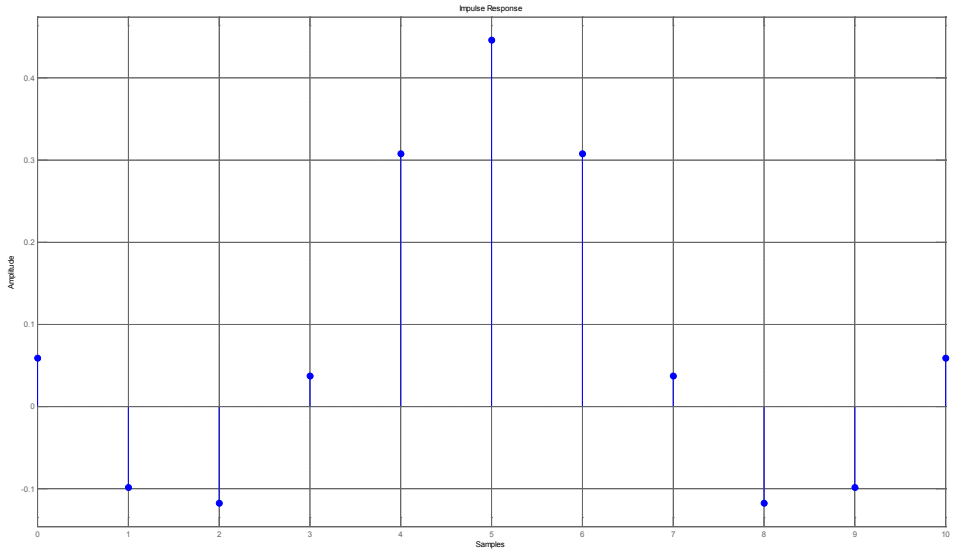
Linear phase FIR filter

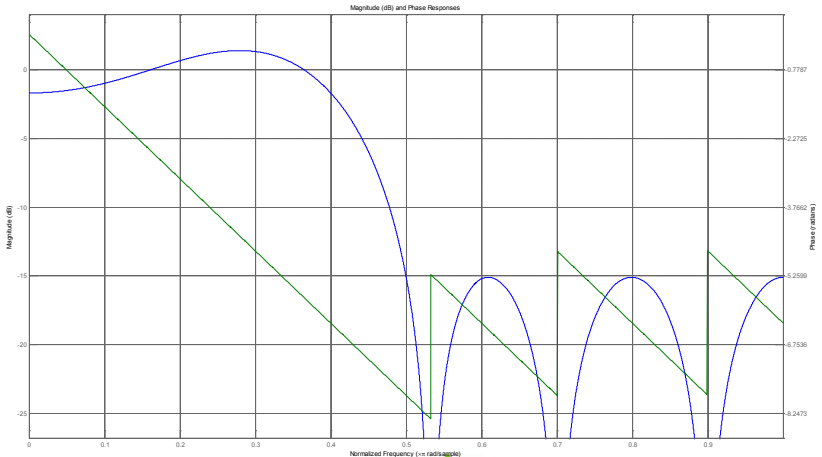
- Type I: h(n) = h(N-n) with even N .
- Type II: h(n) = h(N-n) with odd N.
- Type III: h(n) = -h(N-n) with even N.
- Type IV: h(n) = -h(N-n) with odd N.



Type I
Even order - Even symmetric

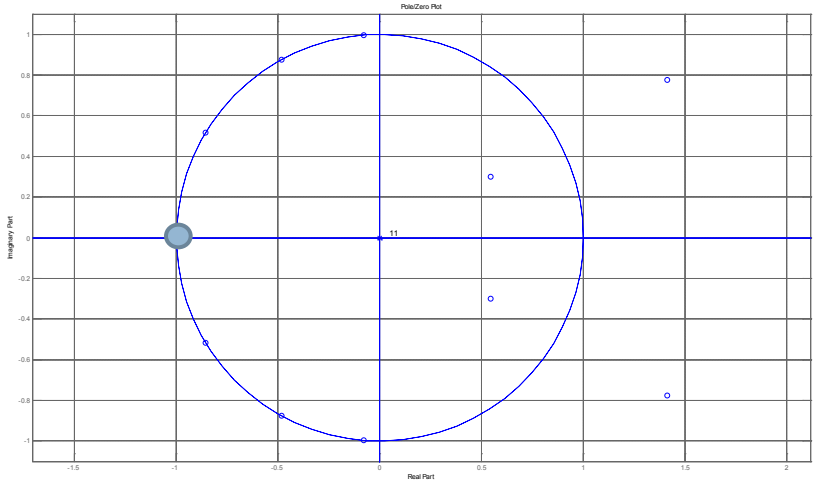


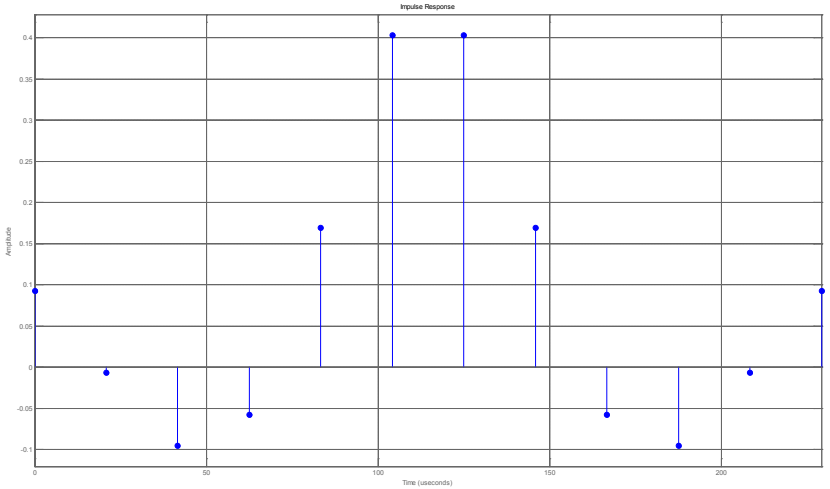




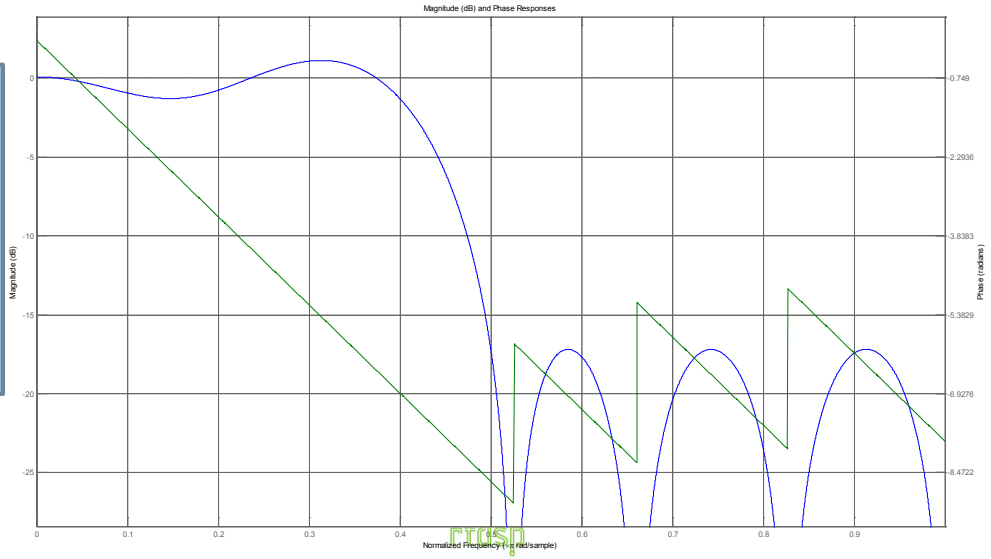
rtdsp

Type II
Odd order – Even symmetric

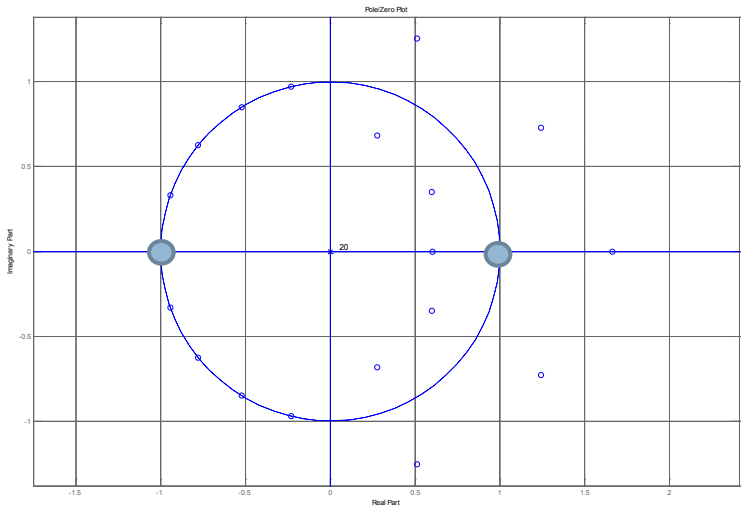


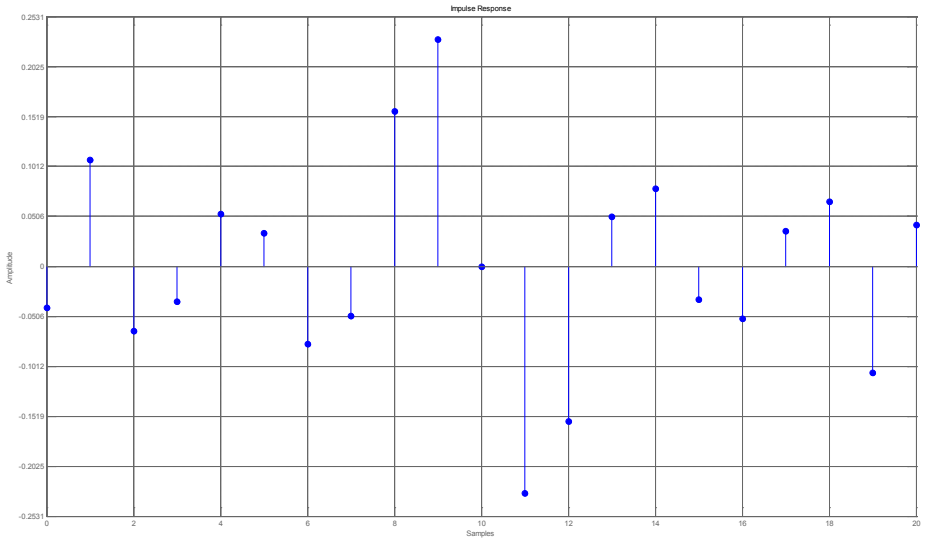


We can´t design high pass filters (HPF)

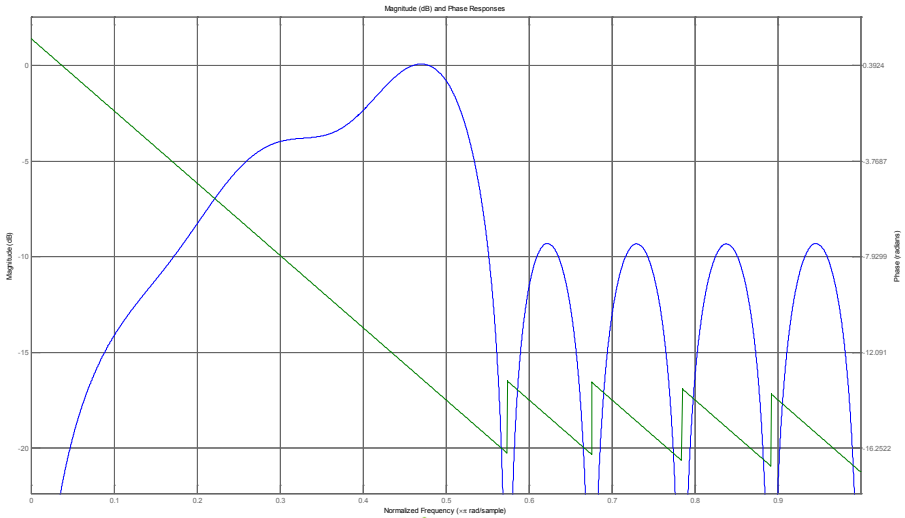


Type III
Even order - Odd symmetric

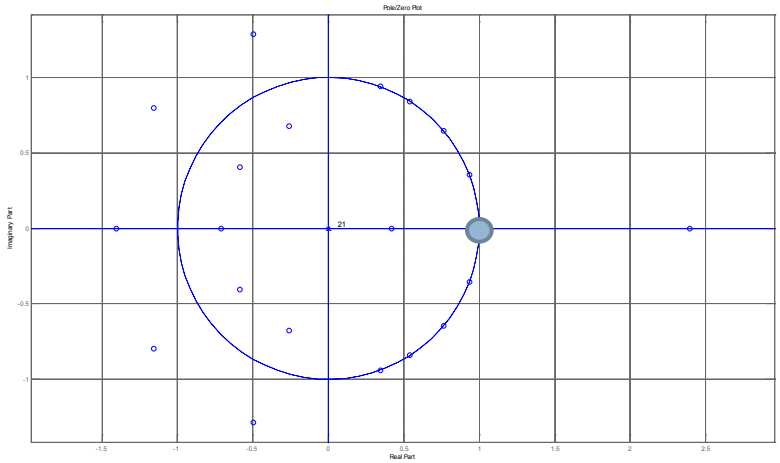


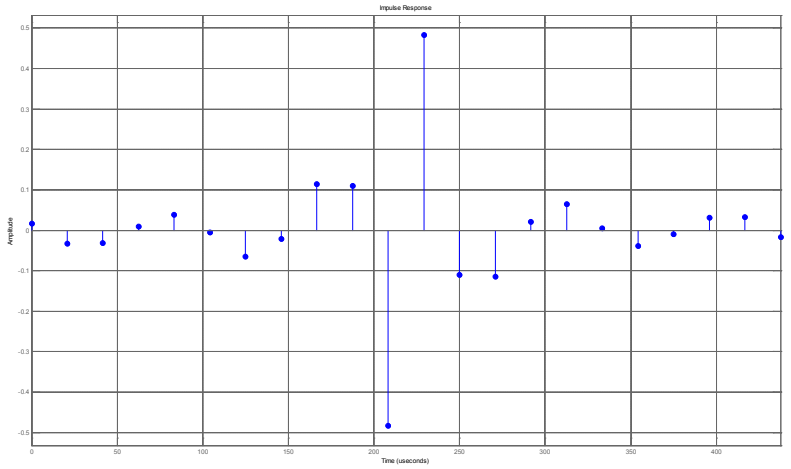


We can´t design low and high pass filters (LPF and HPF)

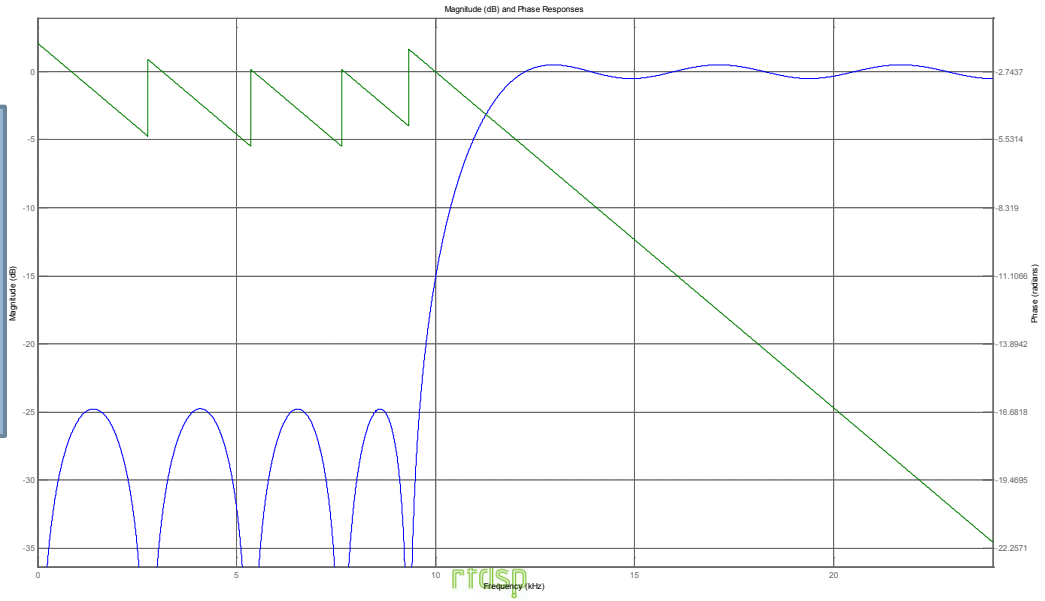


Type IV
Odd order – Odd symmetric

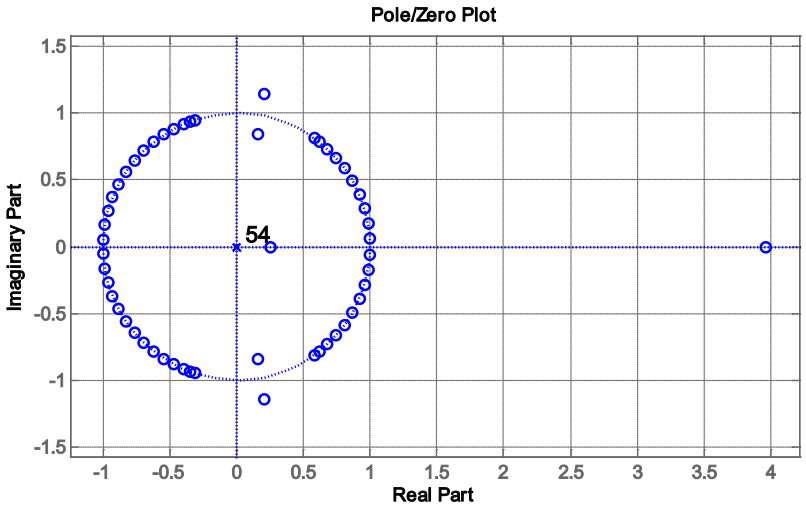


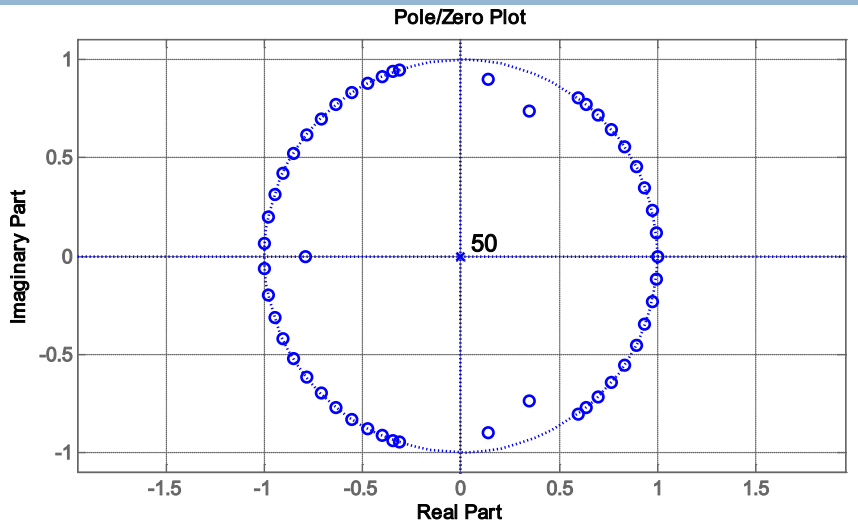


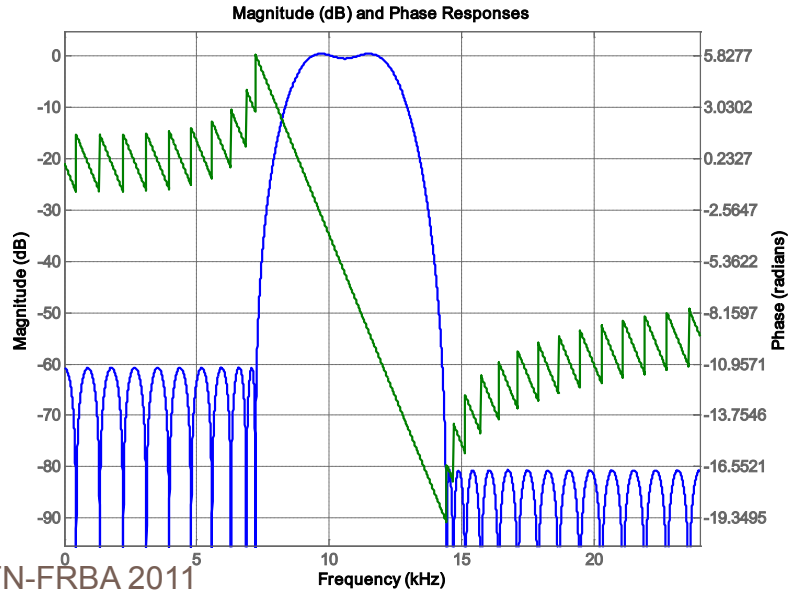
We can´t design low pass filters (LPF)

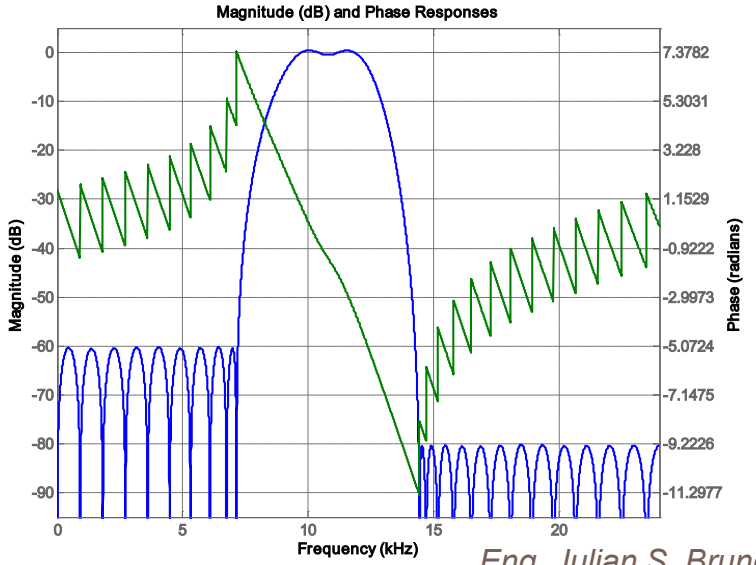


Minimun phase FIR filter

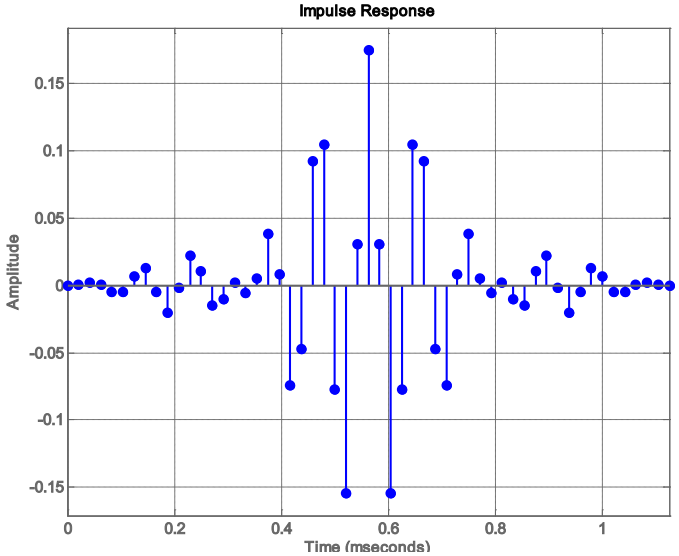


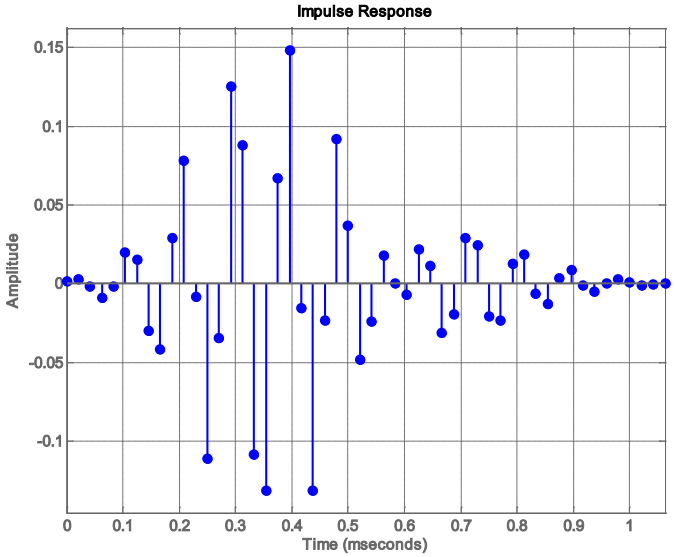


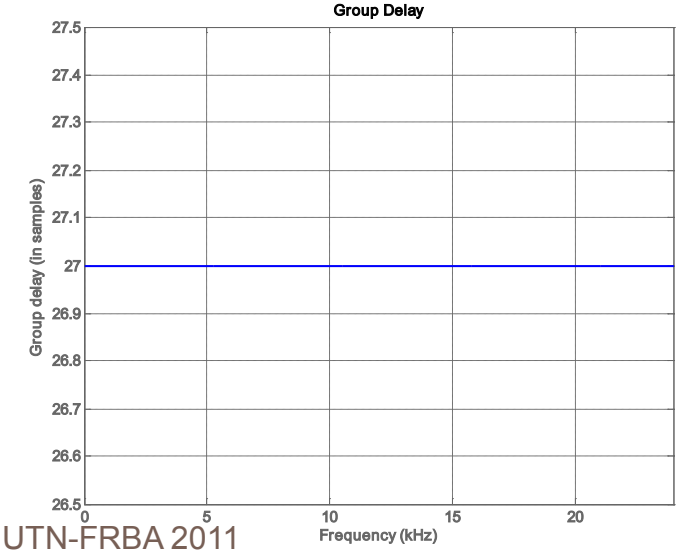


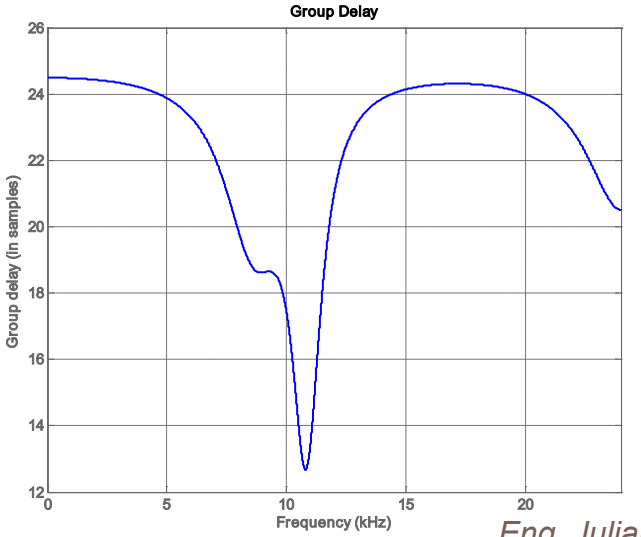


Minimun phase FIR filter









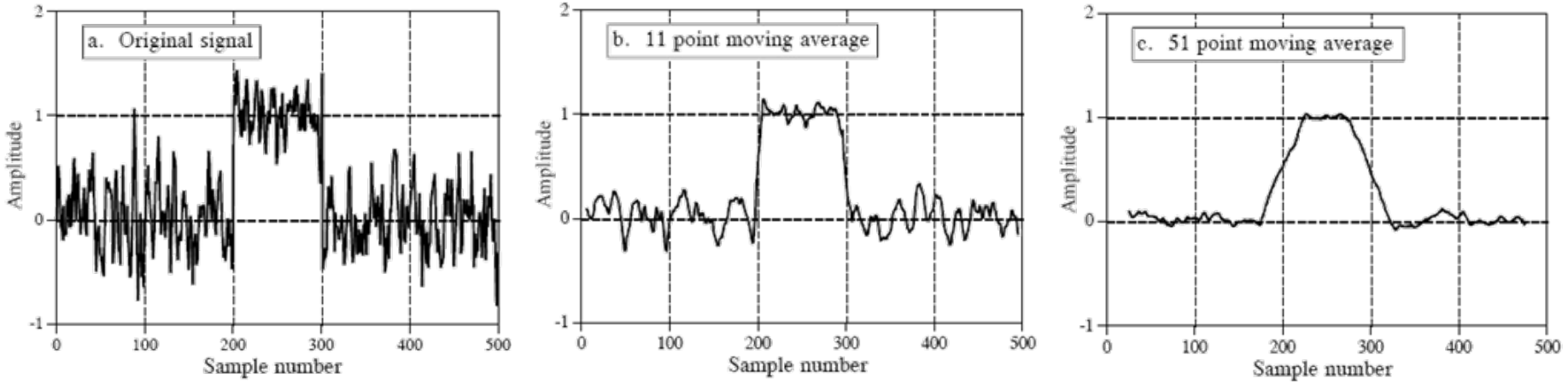
Minimum-Phase vs Linear Phase

- L-P FIR filter has zeros inside, on, and outside the unit circle.
- M-P FIR filter has all of its zeros on or within the unit circle.
- Given optimal M-P and L-P digital FIR filters that meet the ***same magnitude specification***, the M-P filter would have a ***reduced filter length*** *of typically one* half to three fourths of the L-P filter length, and ***Minimum group delay***, *where energy would be concentrated in the* ***low-delay*** instead of the medium-delay coefficients
- M-P filters can simultaneously meet ***constraints on delay and magnitude response*** while generally requiring fewer computations and less memory than L-P filters.

FIR filters: Coefficients calculation

- Moving average (MA)
- Windowed Sinc filters
- Remez exchange (Parks McClellan) method

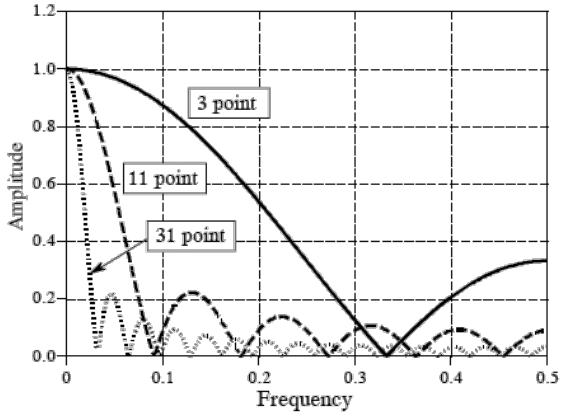
Moving average filters

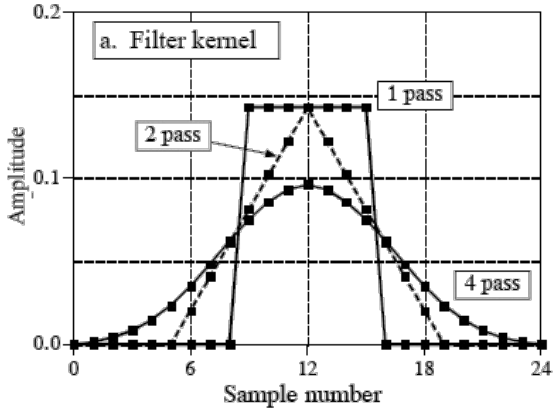


$$y[n] = \frac{1}{M}\sum_{j=0}^{M-1} x[n-j] \quad ; \quad y[n] = \frac{1}{M}\sum_{j=0}^{M-1} x[n-j+M/2]$$

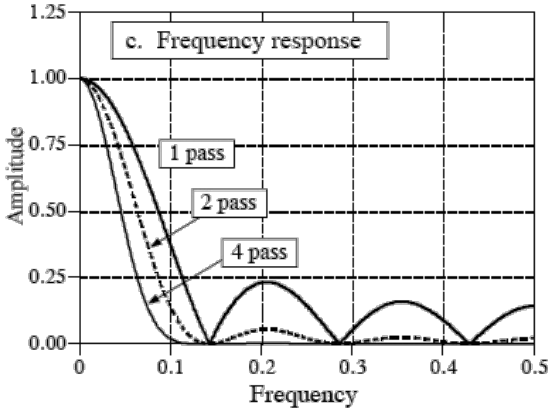
- Each output will be the average of the last M samples of the input.
- This kind of filters is very easy to implement, but it has very poor roll-off and attenuation in stopband.
- It is mainly used for smoothing purposes.

MA filters: applications



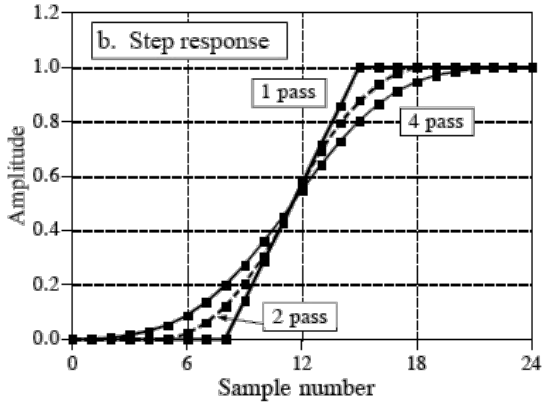


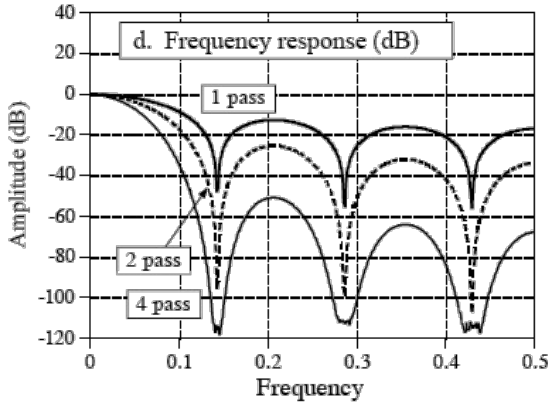
FFT



Integrate

20 Log( )





- Improvements can be achieved if we increase M or if the same filter is passed multiple times.

MA filters: Recursive Implementation

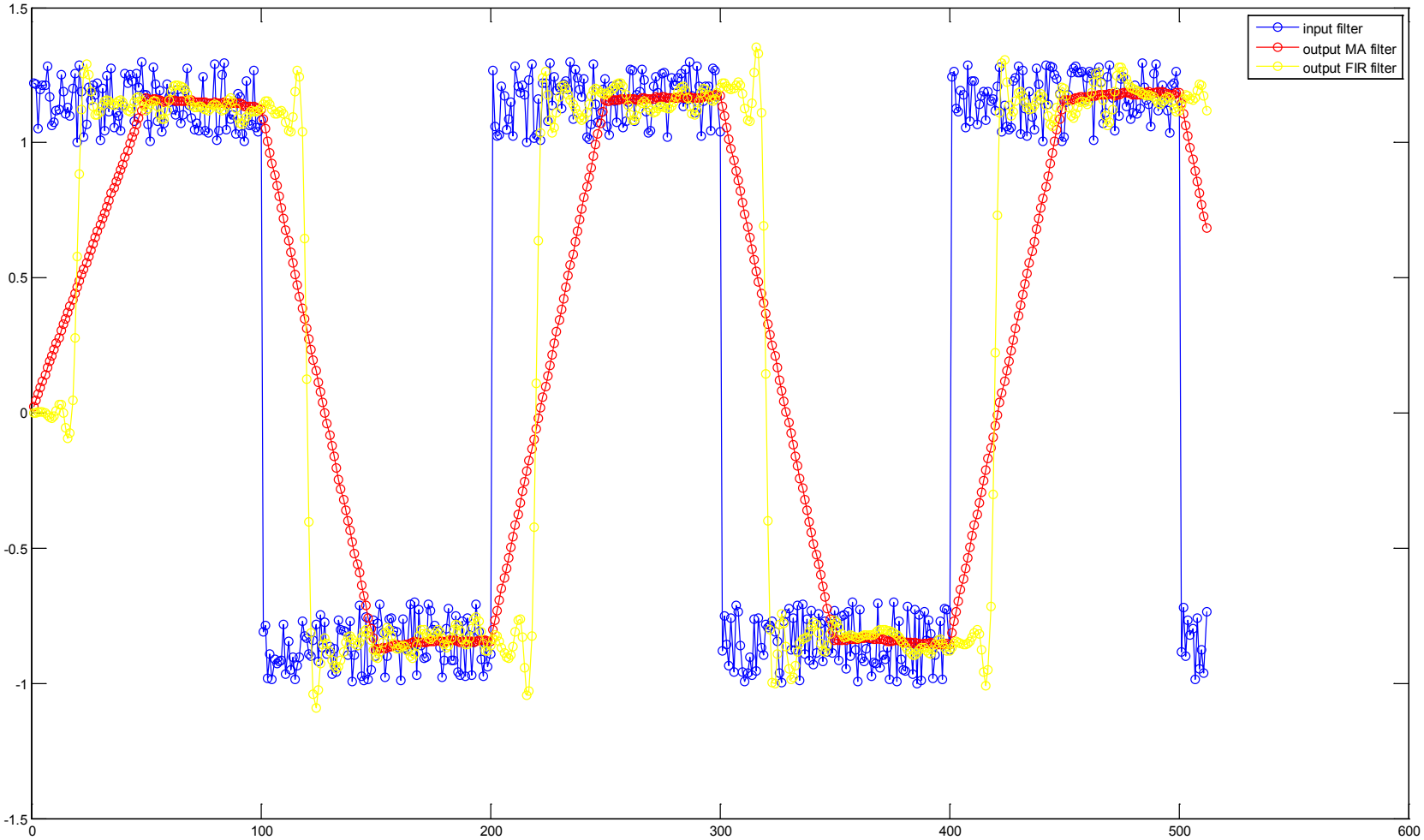
$$y[50] = x[47] + x[48] + x[49] + x[50] + x[51] + x[52] + x[53]$$

$$y[51] = x[48] + x[49] + x[50] + x[51] + x[52] + x[53] + x[54]$$

$$y[51] = y[50] + x[54] - x[47]$$

- The cost of implementing a MA filter is M sums and 1 multiplication.
- This can be improved if a recursive approach is taken.
- So any MA filter can be implemented by 2 sums and one accumulator.
- Keep in mind that a higher M implies a higher delay.

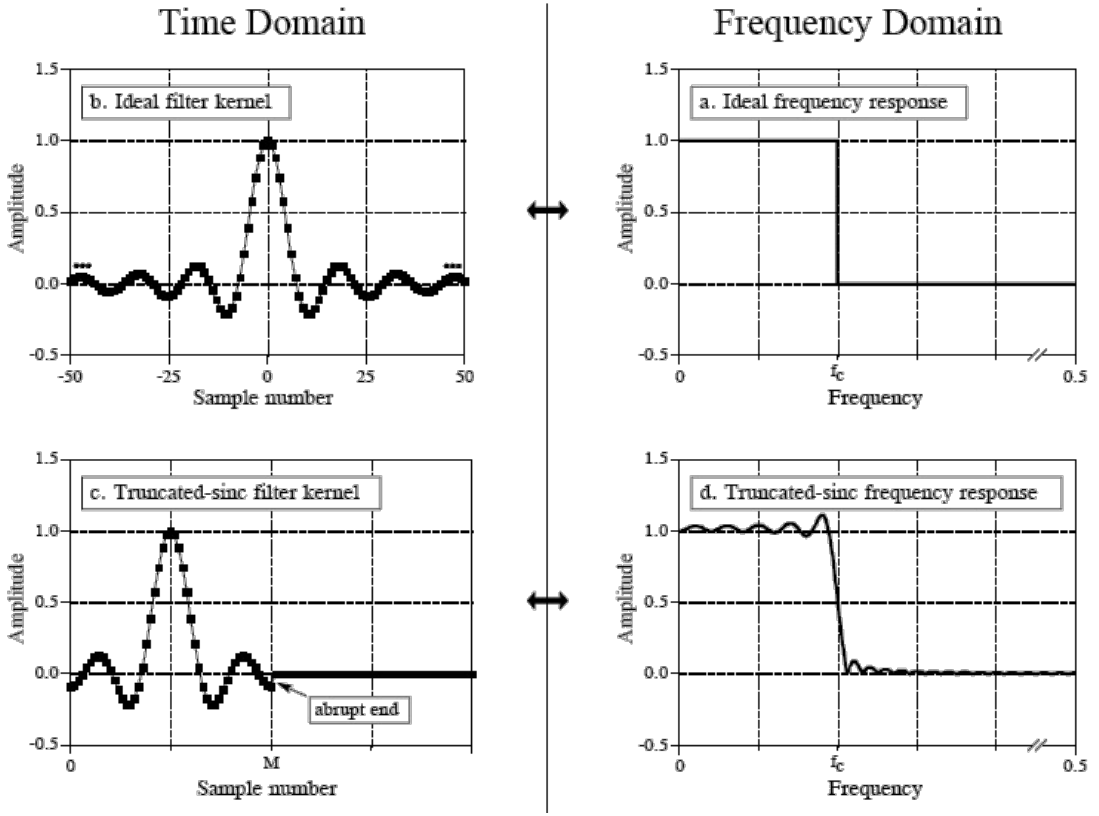
MA Filter: Example



MA filter: M=50 , An=10% , FIR filter: Equiripple order 39

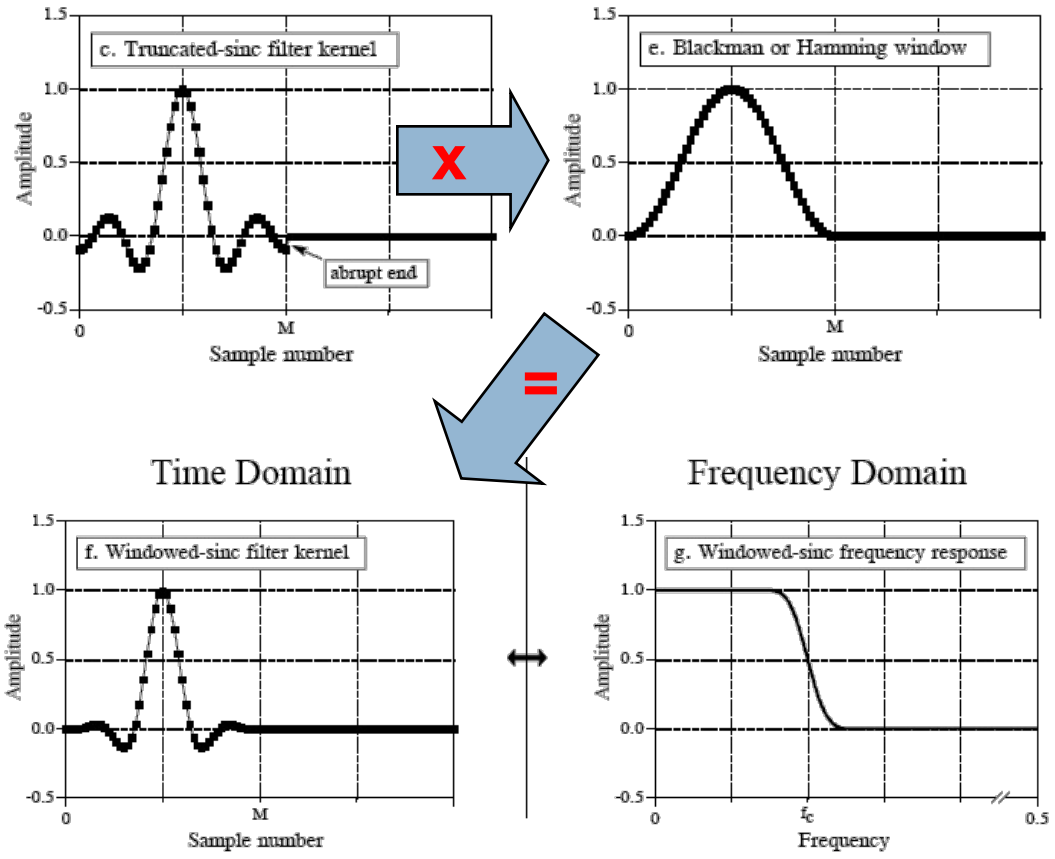
Windowed Sinc FIR filters

- A sinc of infinite duration is the time domain equivalent of an ideal filter response.
- As it is impossible to store this signal, it is truncated to M samples.
- This truncation causes a severe degradation to the ideal frequency response.



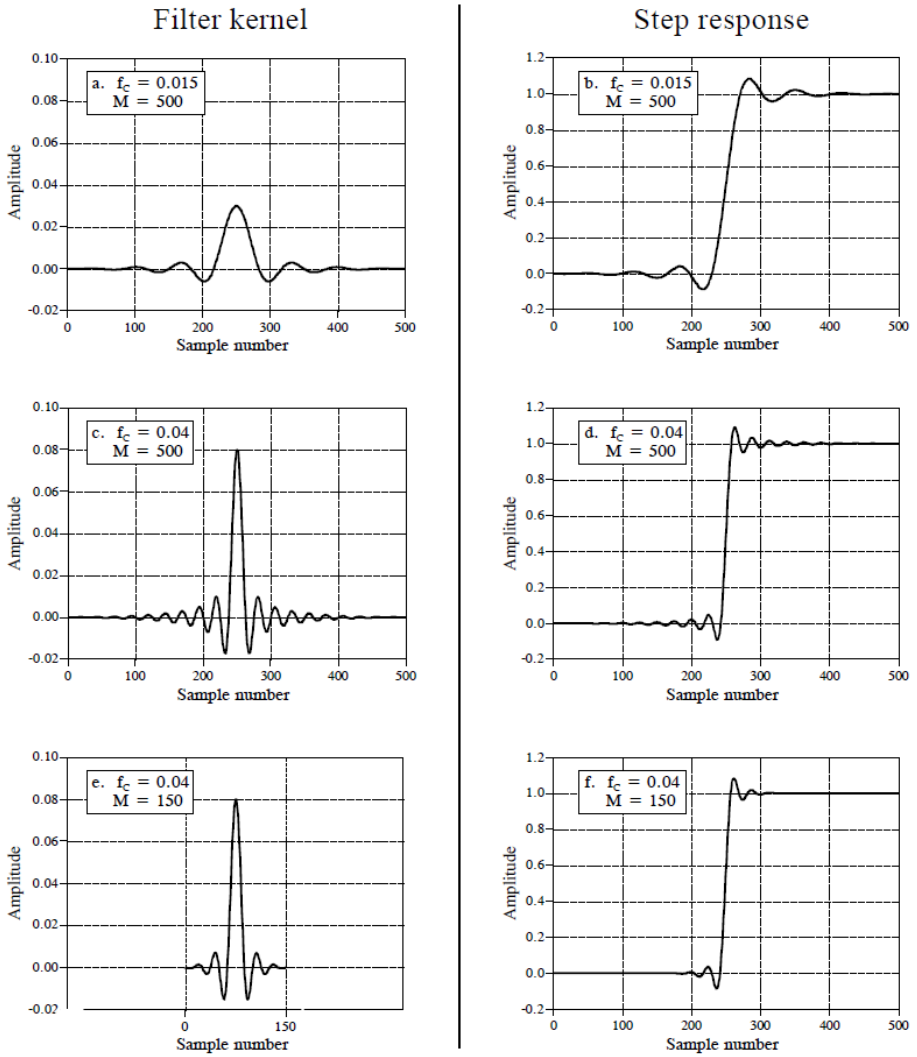
Windowed Sinc FIR filters

- In order to obtain better filter features, the truncated sinc can be multiplied by a window function.
- These windows achieve either better smoothness, roll-off or stopband attenuation than the sinc.
- For this purposes, many windows were designed.



Windowed Sinc FIR filters: Design

- Windowed Sinc filters are very easy to design.
- We only have to multiply a window formula to the sinc signal.
- One problem of window filters is that they are not optimal in the kernel length sense.
- The frequency of the sinusoidal oscillation is approximately equal to Fc

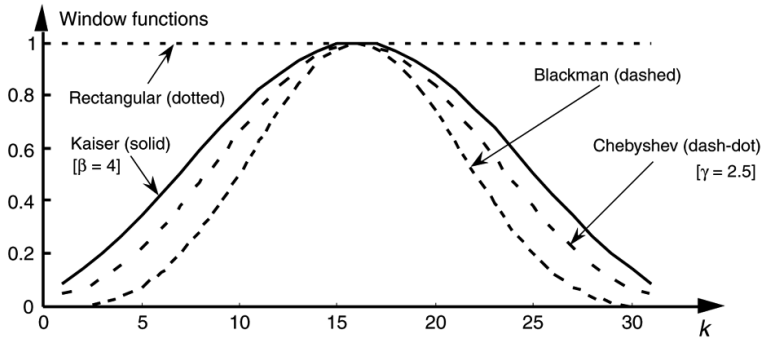


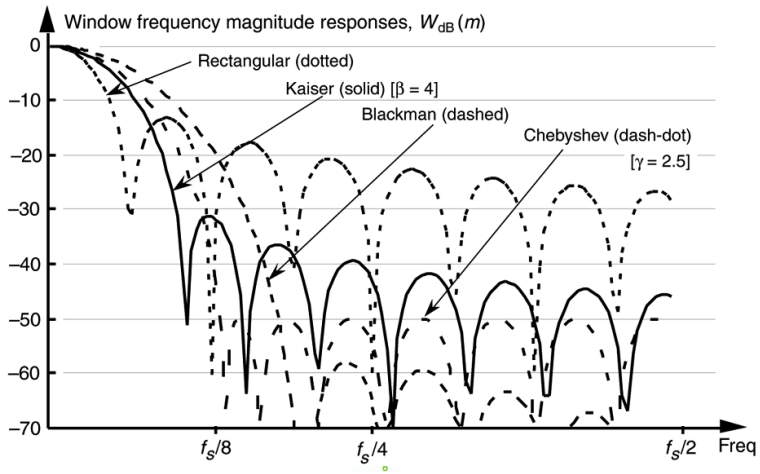
Windowed Sinc FIR filters: Windows

Hamming: $w[k+1] = 0.54 - 0.46\cos\left(2\pi\frac{k}{n-1}\right), \quad k = 0, \ldots, n-1$

Blackman: $w[k+1] = 0.42 - 0.5\cos\left(2\pi\frac{k}{n-1}\right) + 0.08\cos\left(4\pi\frac{k}{n-1}\right), \quad k = 0, \ldots, n-1$

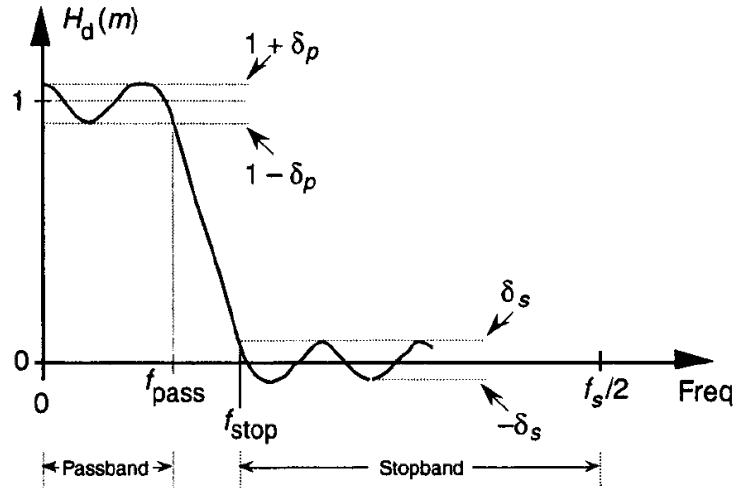
Hanning: $w[k+1] = 0.5\left(1 - \cos\left(2\pi\frac{k}{n-1}\right)\right), \quad k = 0, \ldots, n-1$

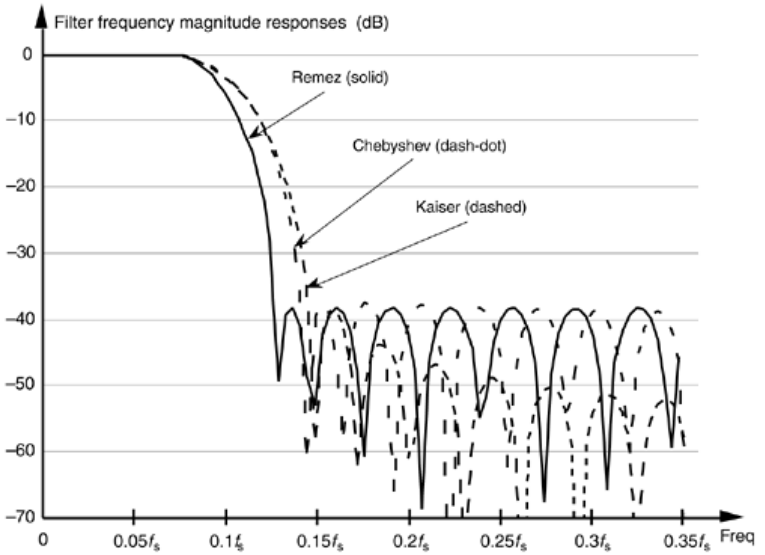




Optimal FIR filters: Remez exchange (Parks McClellan) method

- This design algorithm allows filter designs that are optimal in the kernel length sense.
- The usage of the method is as simple as passing the filter specification to the algorithm and receiving its filter kernel.

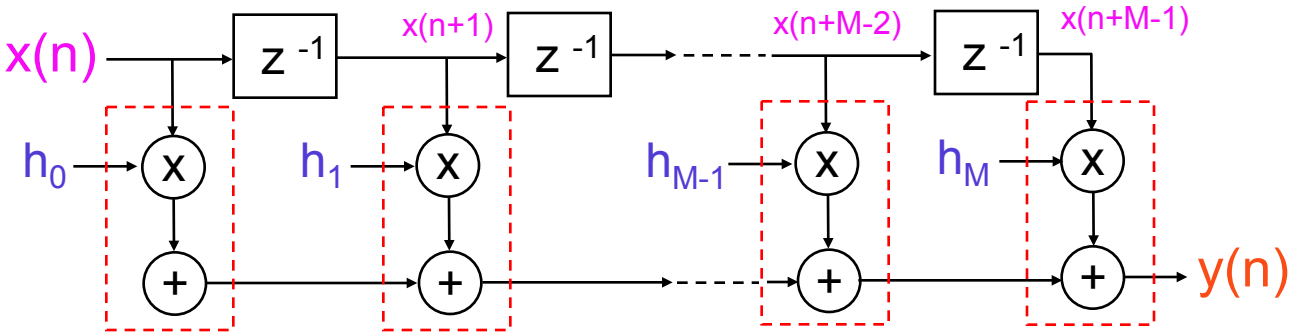




FIR filters: Implementation

- Direct form or transversal structure.
- Symmetry (Linear phase) optimization.
- Fast convolution (overlap add/save).

FIR filters: Direct or transversal form

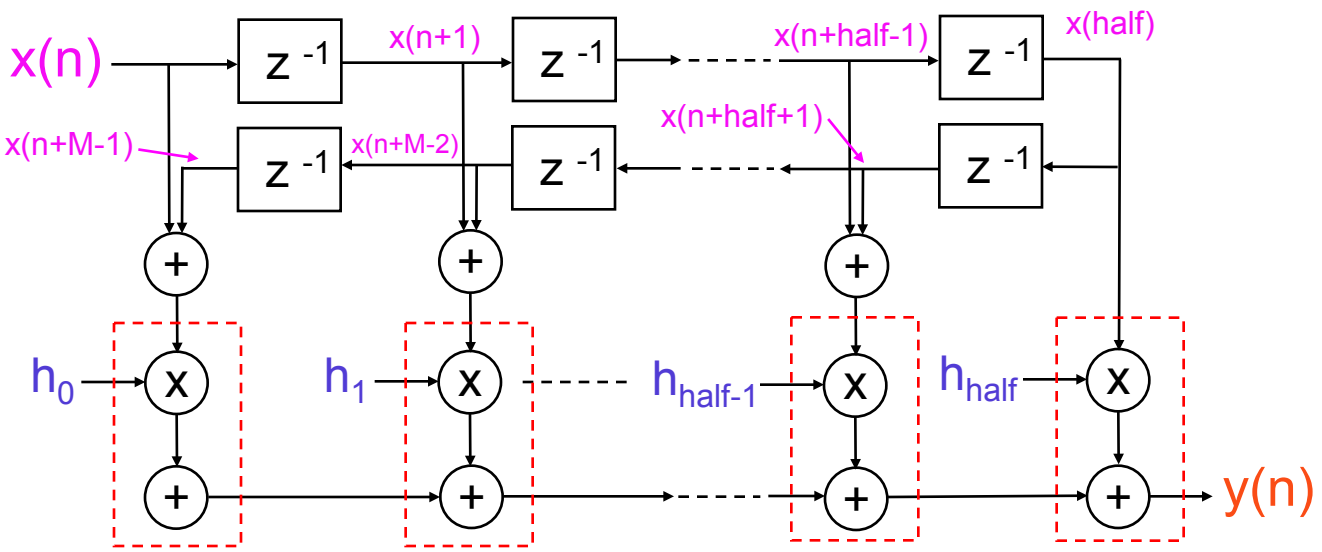


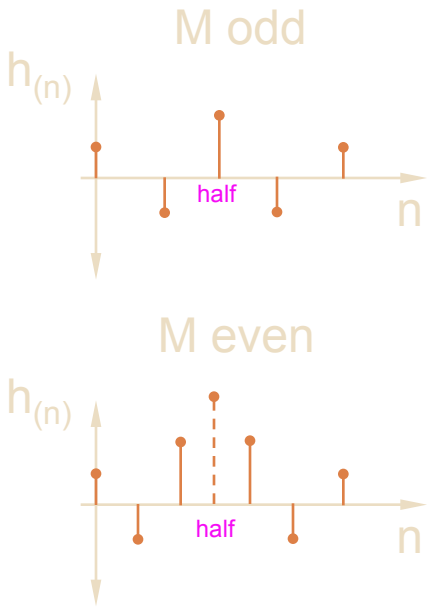
Computational cost:

- M MAC operations
- M RAM positions (filter)
- M RAM positions (signal)

- Most DSP architectures are optimized for implementing this structure.

FIR filters: Linear phase form

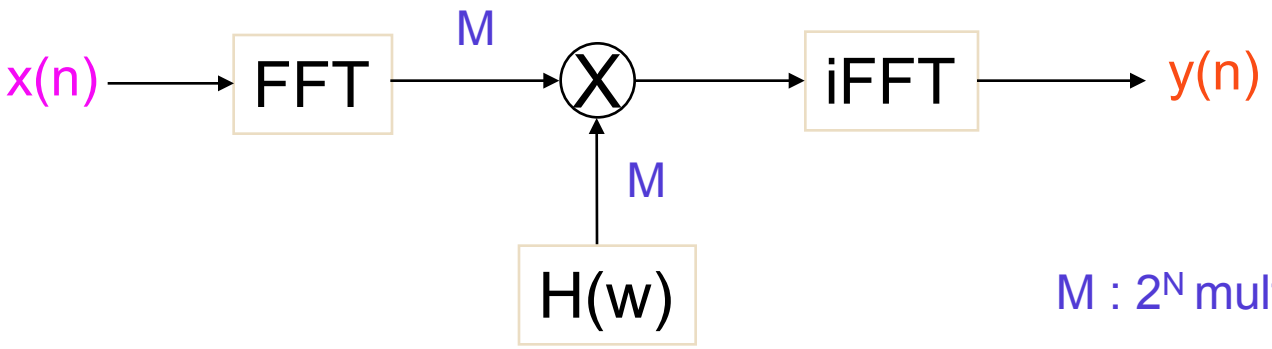




Computational cost:

- M/2 (even) M/2+1 (odd) MAC operations
- M/2 (even) M/2+1 (odd) RAM positions (filter)
- M RAM positions (signal)

FIR filters: Fast convolution



Computational cost:

- 2 FFT (M . $\log_2$(M))
- M/2 complex multiplications
- M RAM positions (filter)
- M (2M for performance) RAM positions (signal)

- Overlap add/save algorithm.

FIR filters: Wordlength effects

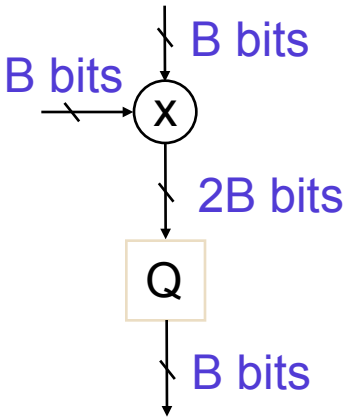
1. ADC quantization noise.

$$SNR_{ADC} = 1.76 + 6.02 \,.\, ADC_{Bits}$$

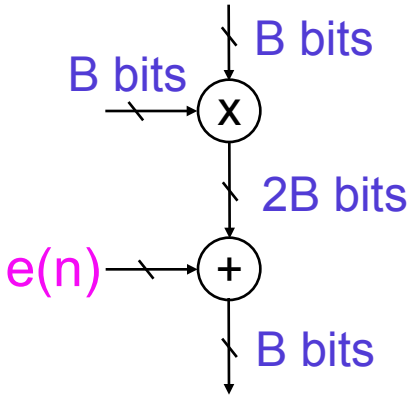
2. Coefficients quantization noise.
3. Roundoff quantization noise.
4. Arithmetic overflow noise.

FIR filters: Roundoff quantization

Non linear model



Linear model



$$\sigma_Q^2 = \frac{q^2}{12}$$ (roundoff noise power)

q : ADC quantization step (V)

- Each quantization process (multiplication) degrades system's noise figure.

Infinite Impulse Response

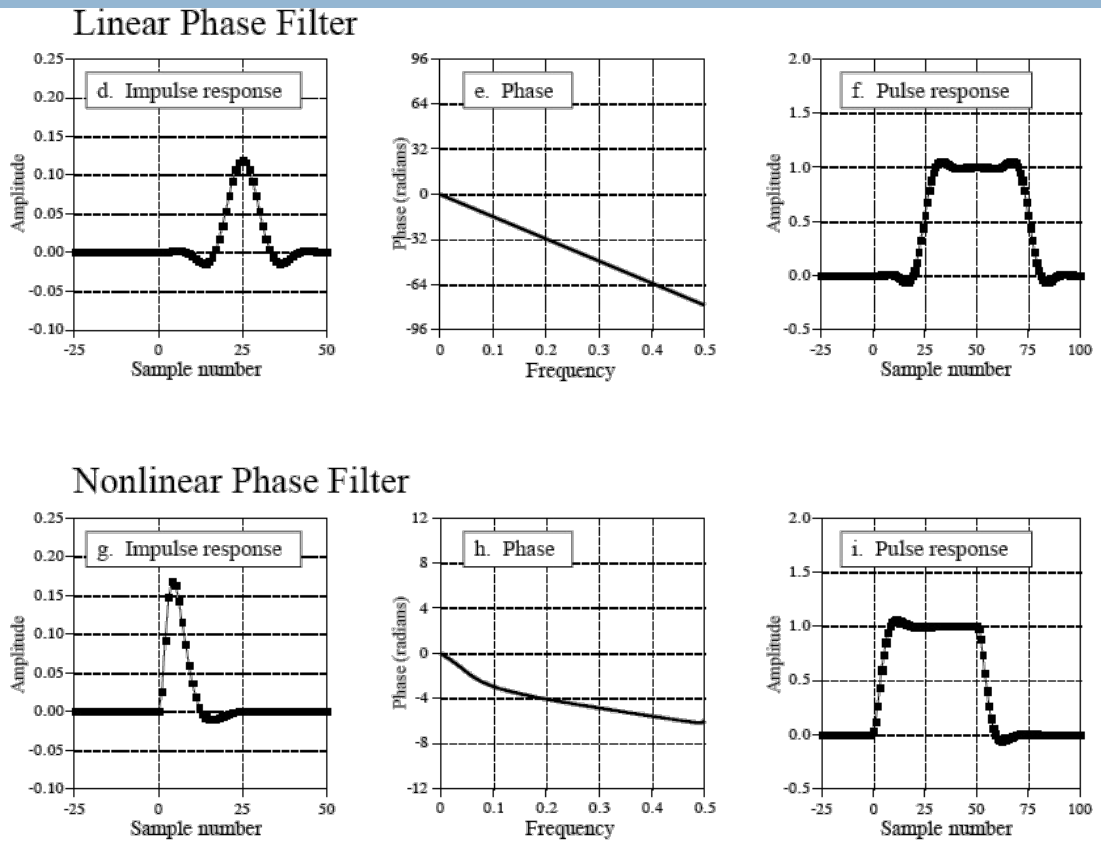
IIR

Eng. Julian S. Bruno

IIR filters: General characteristics

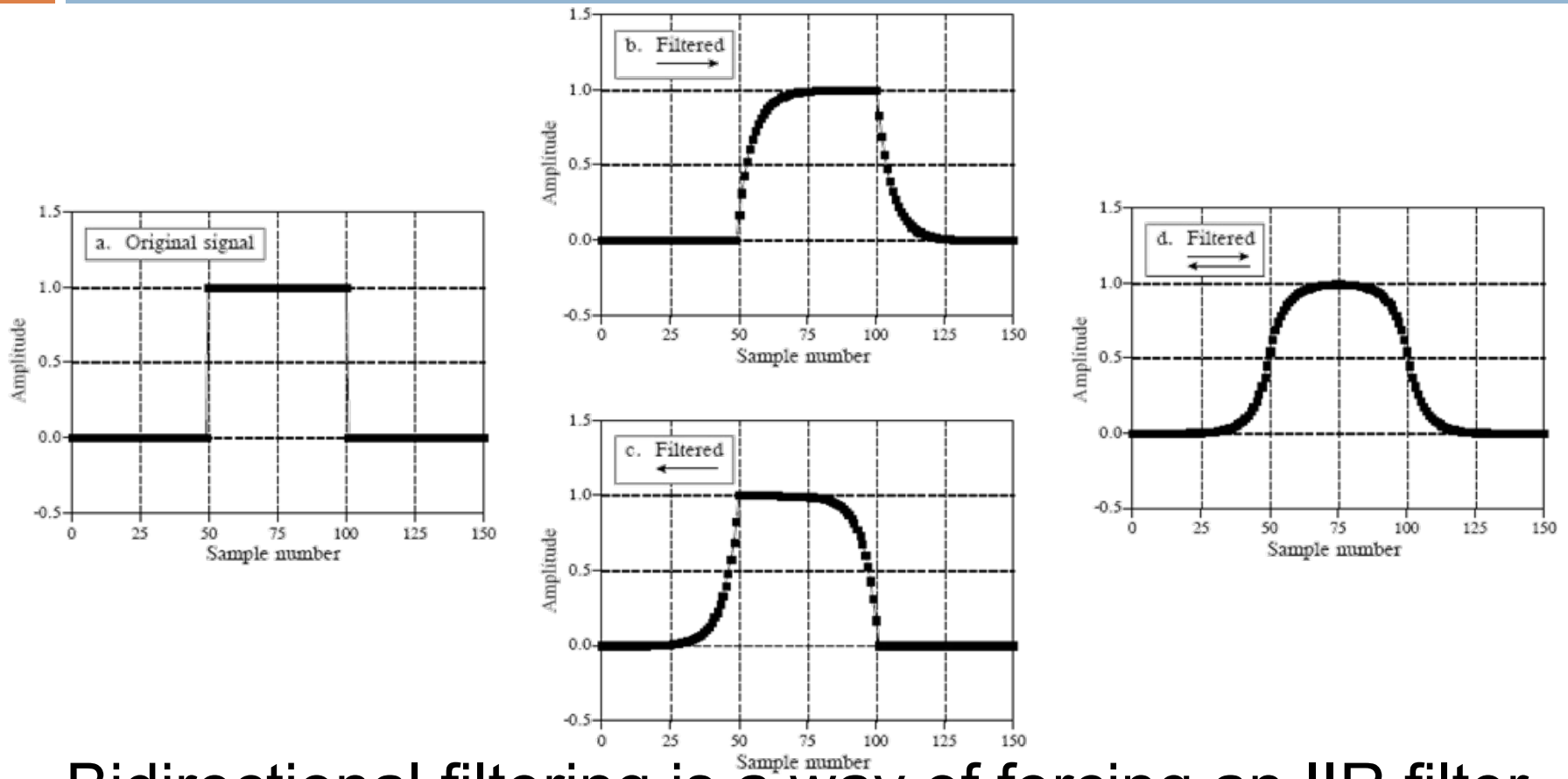
- IIR filter output sample ***depends*** on previous ***input*** samples and previous filter ***output*** samples.
- Implemented using a recursion equation, instead a convolution and ***don´t have linear phase responses***.
- IIR filters have ***poles*** and ***zeros around the Z plane***, they ***are not always stable***.
- Finite duration of nonzero input values / Infinite duration of nonzero output values. ***Infinite Impulse Response***
- IIR filters achieves better performance than FIR for the same kernel length.
- Implementation is more complicated, and requires floating point or large word lengths.

IIR filters: Non linear phase response



- As in analog domain, linear phase could be achieved at expense of loosing other desirable features.
- In other words, IIR filters are much faster than FIR filters but filter stability and phase distortion is a VERY important issue.

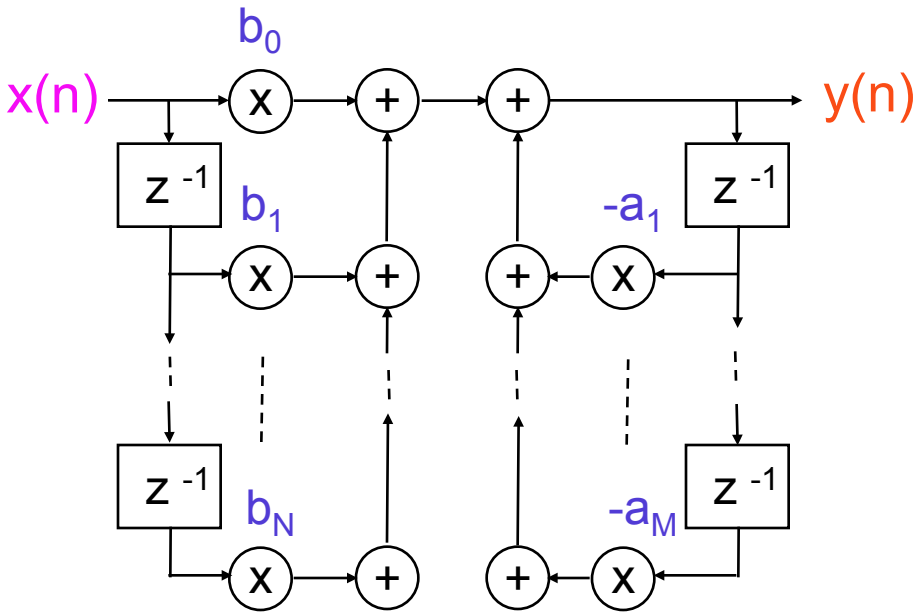
IIR filters: Linear phase with bidirectional filtering

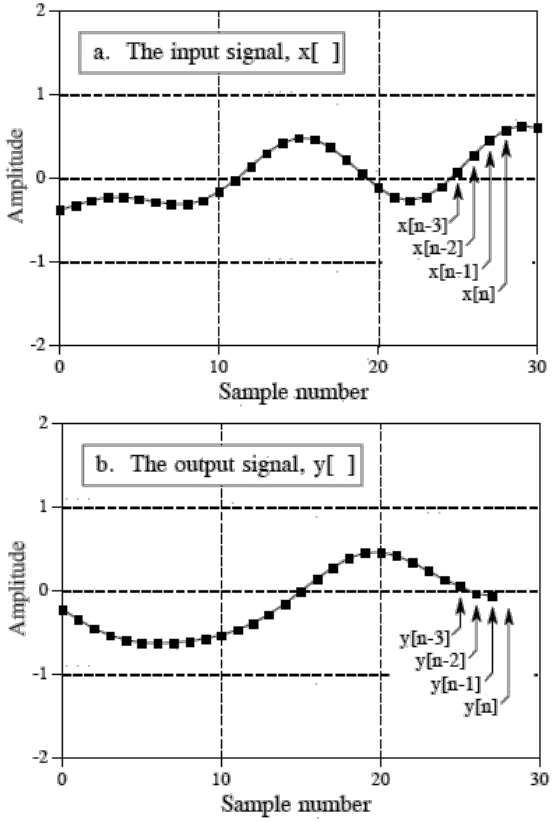


- Bidirectional filtering is a way of forcing an IIR filter to have perfect linear phase, at expense of speed since, at least, the computing cost is duplicated.

IIR filters: the recursion equation

$$y(n) = \sum_{k=0}^{N} b_{(k)}.x(n-k) - \sum_{k=1}^{M} a_{(k)}.y(n-k)$$





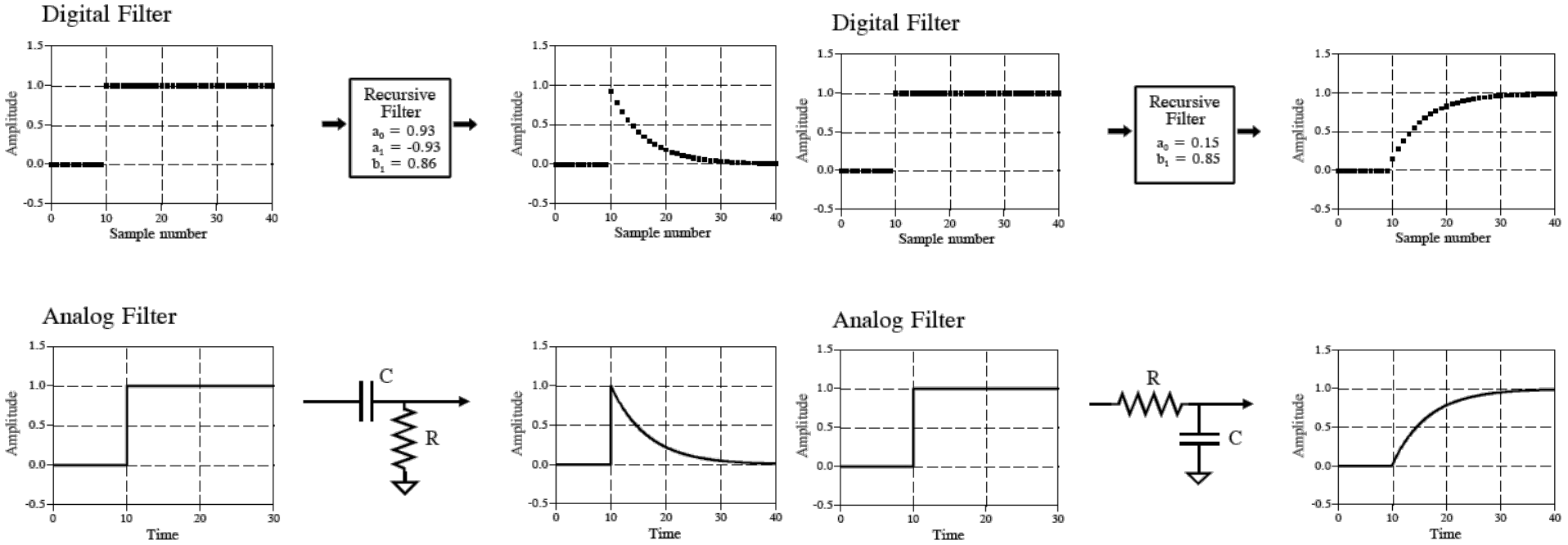
- Each output is a linear combination of past input and output samples

IIR filters: Coefficients calculation

- Pole-zero placement.
- Impulse invariance.
- Matched z-transform.
- Bilinear z-transform.

The Bad news is that there´s no direct method for computing the IIR filter's coefficients from the impulse response!!!!

IIR filters: Analog & Digital



For HP: $a_0 = (1+x)/2$, $a_1 = -(1+x)/2$, $b_1 = x$

And for LP: $a_0 = 1-x$, $b_1 = x$

And for both: $x = e^{-1/d}$ (Time decay)

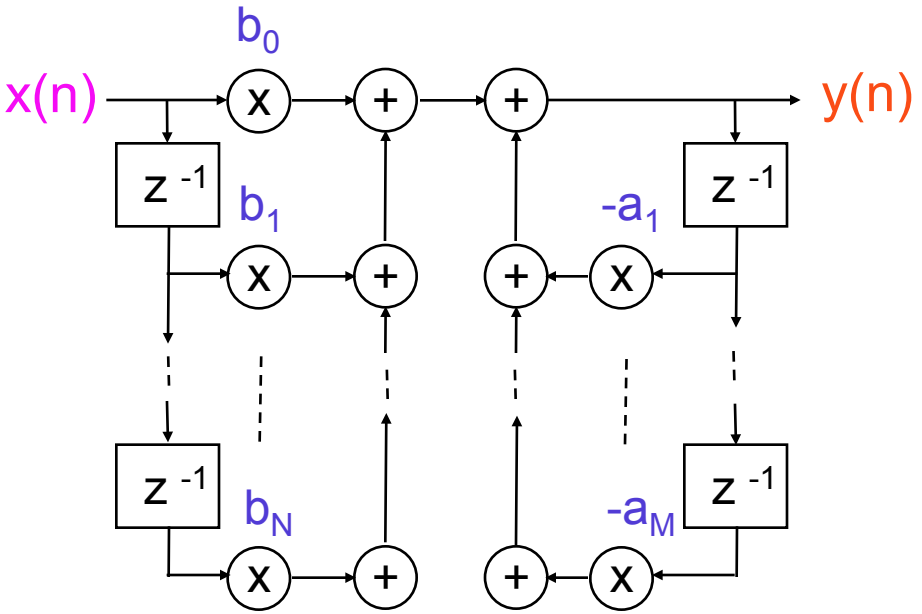
$x = e^{-2\pi f_c}$ (Cutoff frequency)

- As in the analog domain, simple filters could be implemented with a few coefficients.

IIR filters implementation: Direct Form 1

- DF1 is the most direct way of implementing an IIR filter.
- It uses two buffers (delay lines) and 2(M+1) multiplications and sums.
- More sophisticated forms optimizes memory usage, coefficients precision sensibility and stability.

$$y(n) = \sum_{k=0}^{N} b_{(k)}.x(n-k) - \sum_{k=1}^{M} a_{(k)}.y(n-k)$$

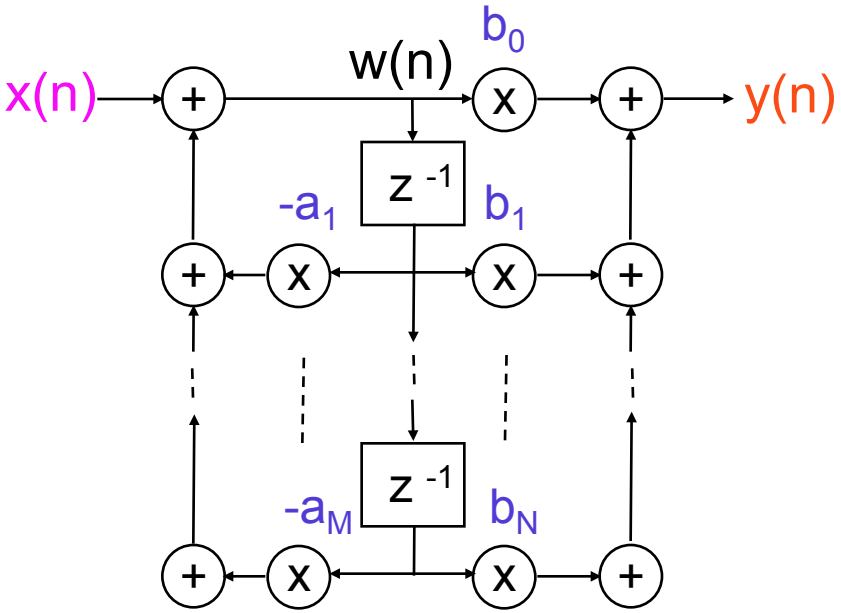


IIR filters implementation: Direct Form 2 (canonic)

- DF2 Reduces memory usage by two, since the delay line is shared.
- Keep in mind that coefficients in DF1 and DF2 are not the same, since they doesn't affect the same signals.

$$w(n) = x(n) - \sum_{k=1}^{M} a_{(k)}.w(n-k)$$

$$y(n) = \sum_{k=0}^{N} b_{(k)}.w(n-k)$$

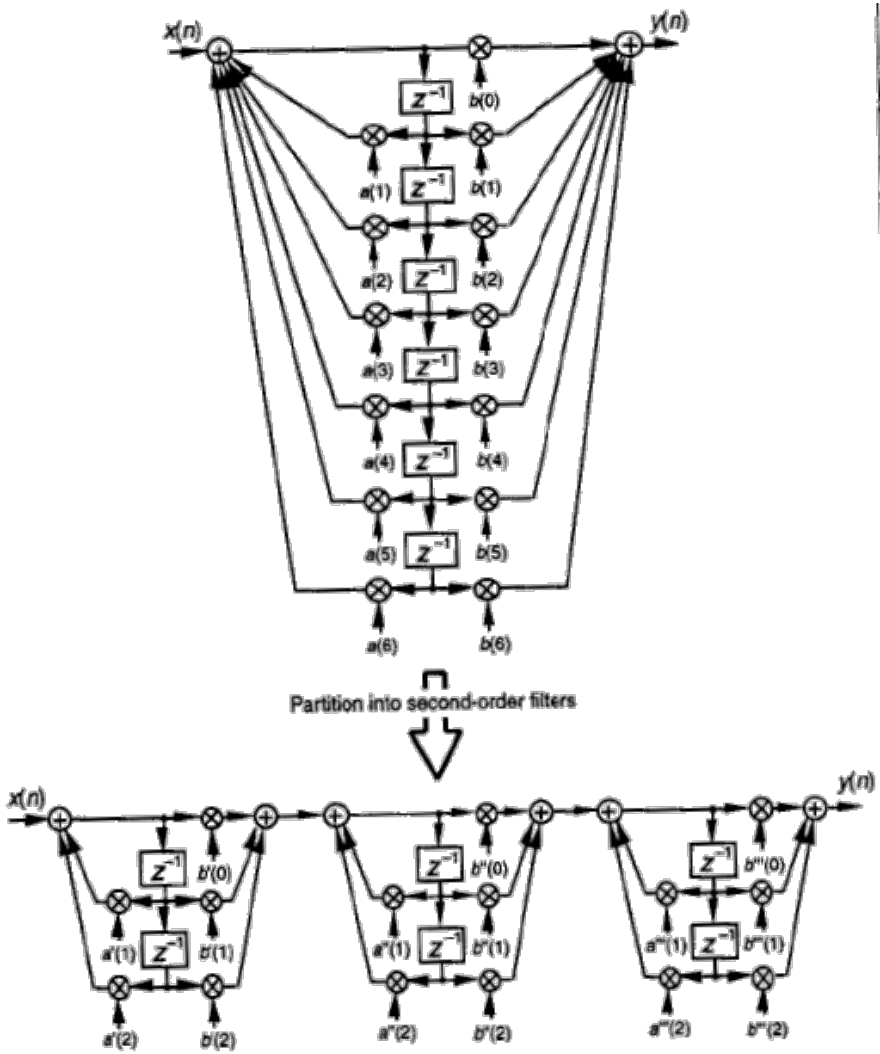


IIR filters implementation: Design Pitfalls

- IIR implementation is conditioned mainly by **finite wordlength effects**.
- Coefficient quantization, overflow and roundoff errors are most common problems.
- To go through these problems, some precautions must be taken into account.

IIR filters implementation: Coefficient quantization sensibility

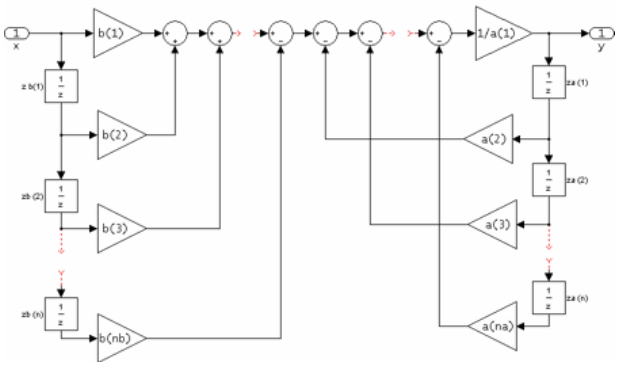
- Frequency response are severely affected for small wordlengths.
- The sensibility grows with the order of the system.
- Second order partitioning mitigate this problem by diminishing speed.
- Instability can occur for very sharp responses.



IIR filters implementation: Coefficient quantization sensibility

$$y(z) = x(z)\sum_{k=0}^{M} b_k z^{-k} - y(z)\sum_{k=1}^{N} a_k z^{-k} \quad , \quad H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}}$$

$$H_q(z) = \frac{\sum_{k=0}^{M} b_{q_k} z^{-k}}{1 + \sum_{k=1}^{N} a_{q_k} z^{-k}} \qquad \begin{aligned} b_{q_k} &= b_k + \Delta b_k \quad k = 1,2,\ldots,M \\ a_{q_k} &= a_k + \Delta a_k \quad k = 1,2,\ldots,N \end{aligned}$$



$$H(z) = \frac{N(z)}{D(z)} \quad , \quad D(z) = 1 + \sum_{k=1}^{N} a_k z^{-k} = \prod_{k=1}^{N} (1 - p_k z^{-1})$$

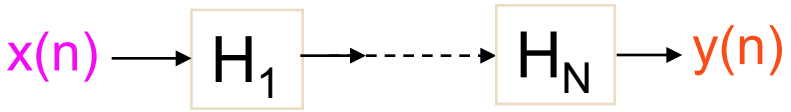
$$H_q(z) = \frac{N_q(z)}{D_q(z)} \quad , \quad D_q(z) = 1 + \sum_{k=1}^{N} a_{q_k} z^{-k} = \prod_{k=1}^{N} (1 - p_{q_k} z^{-1})$$

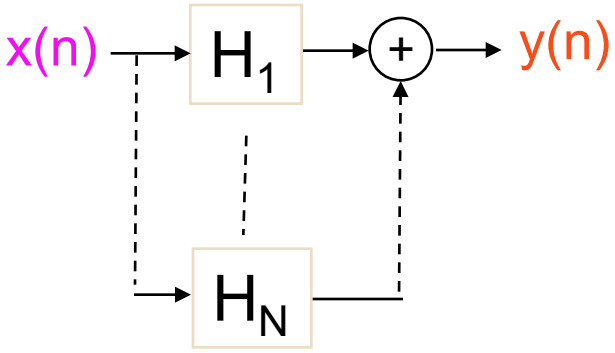
$$p_{q_k} = p_k + \Delta p_k \quad , \quad \Delta p_i = \sum_{k=1}^{N} \frac{\partial p_i}{\partial p_k} \Delta a_k$$

$$\Delta p_i = -\sum_{k=1}^{N} \frac{p_i^{N-k}}{\prod_{\substack{l=1 \\ l \neq i}}^{N} (p_i - p_l)} \Delta a_k$$

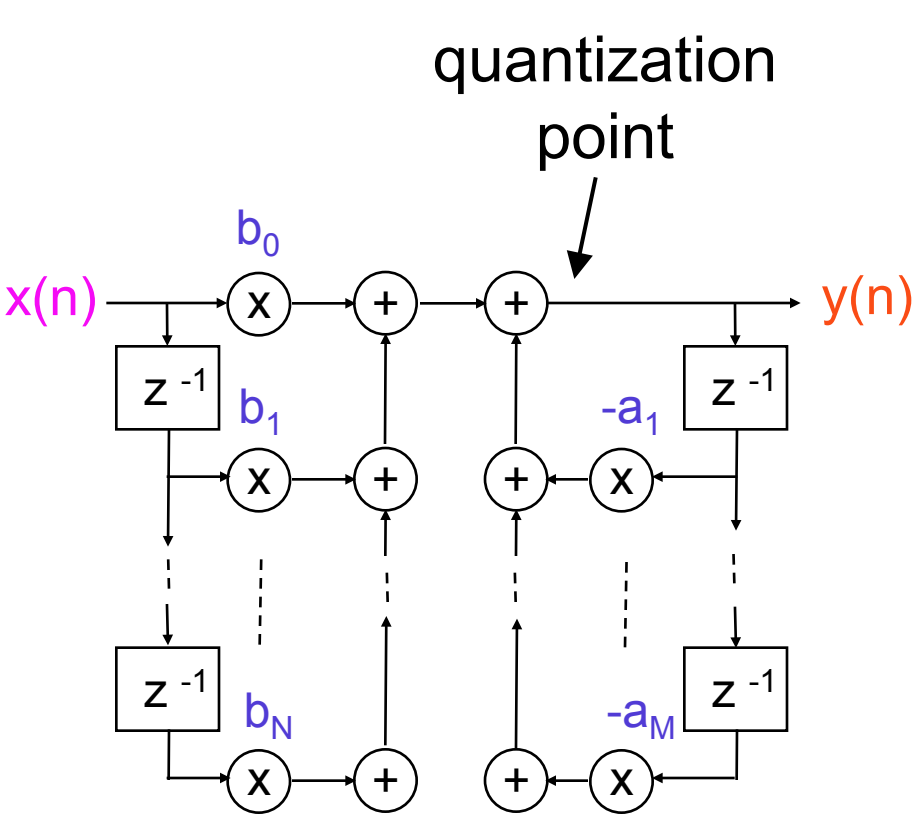
IIR filters implementation: Coefficient quantization sensibility

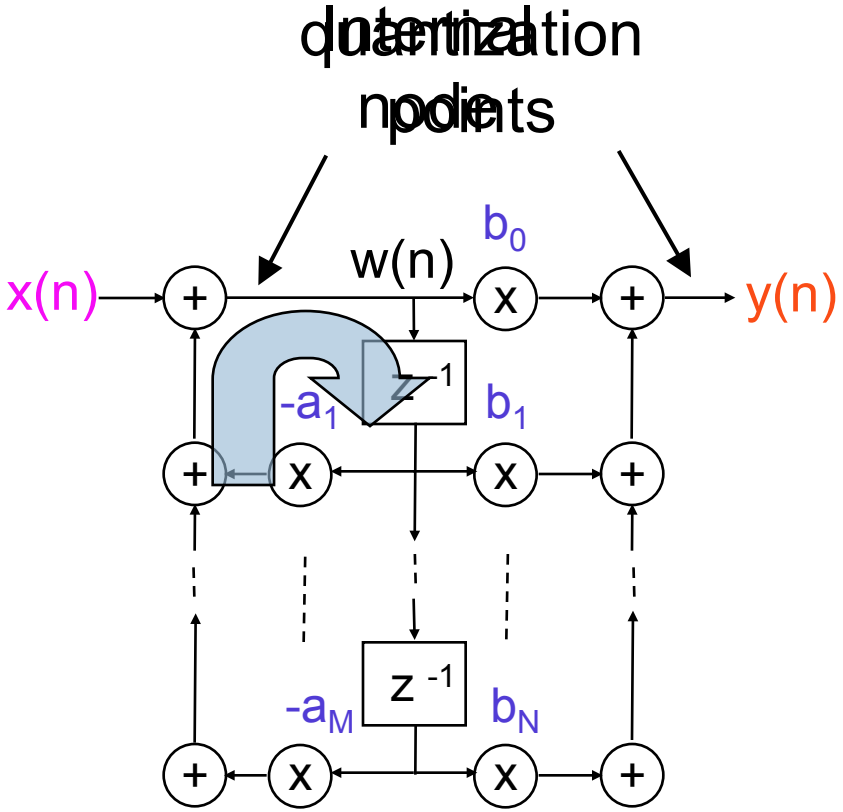
- Second order sections (SOS) can be grouped in cascade or parallel.
- When cascading SOS's the order is chosen to maximize SNR.
- Cascade is the most typically used in DSP processors.





IIR filters: SNR in DF1 and DF2





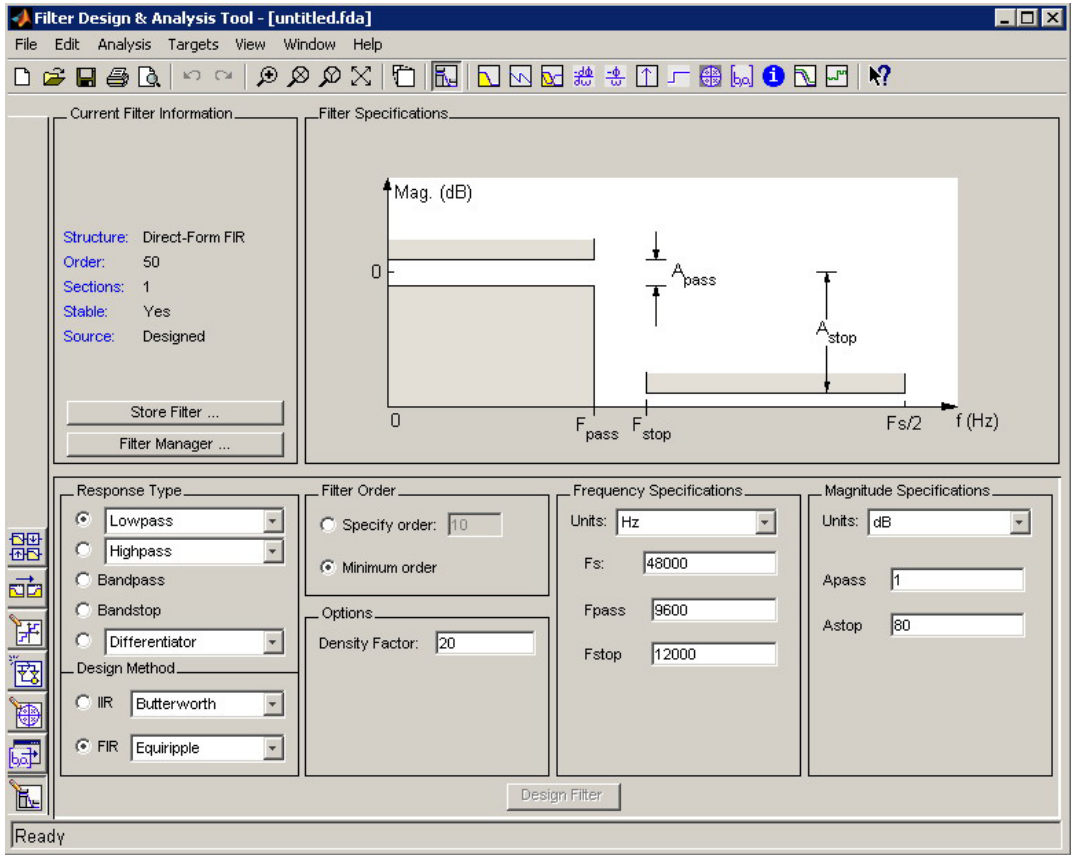
FIR and IIR: comparison chart

Characteristic	IIR	FIR (nonrecursive)
Number of necessary multiplications	Least	Most
Sensitivity to filter coefficient quantization	Can be high.[*] (24-bit coefficients needed for high fidelity audio)	Very low (16-bit coefficients satisfy most FIR filter requirements)
Probability of overflow errors	Can be high[*]	Very low
Stability	Must be designed in	Guaranteed
Linear phase	No	Guaranteed [**]
Can simulate prototype analog filters	Yes	No
Required coefficient memory	Least	Most
Hardware filter control complexity	Moderate	Simple
Availability of design software	Good	Very good
Ease of design, or complexity of design software	Moderately complicated	Simple
Difficulty of quantization noise analysis	Most complicated	Least complicated
Supports adaptive filtering	Yes	Yes

[*] These problems can be minimized though cascade or parallel implementations.
[**] Guaranteed so long as the FIR coefficients are symmetrical

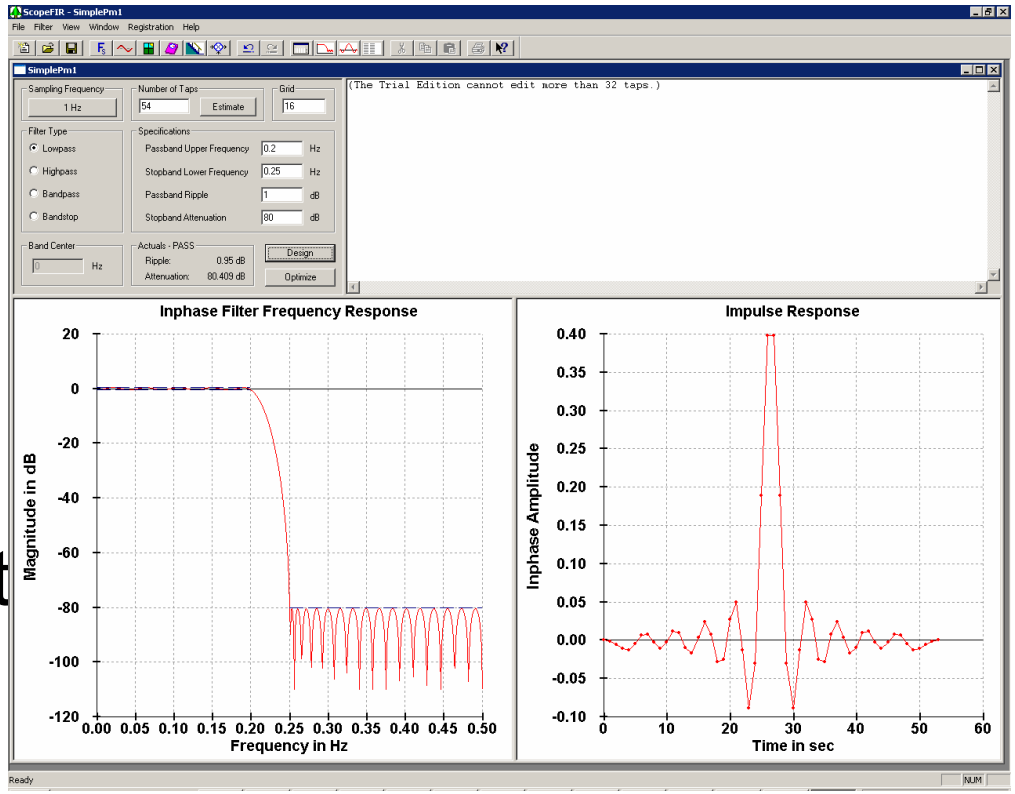
Filters design tools: fdatool

- Matlab's toolbox fdatool allows almost any kind of filter design and is the most sophisticated tool.
- It´s price (Matlab and signal processing toolbox) is aprox. Us$5000 for industrial use.

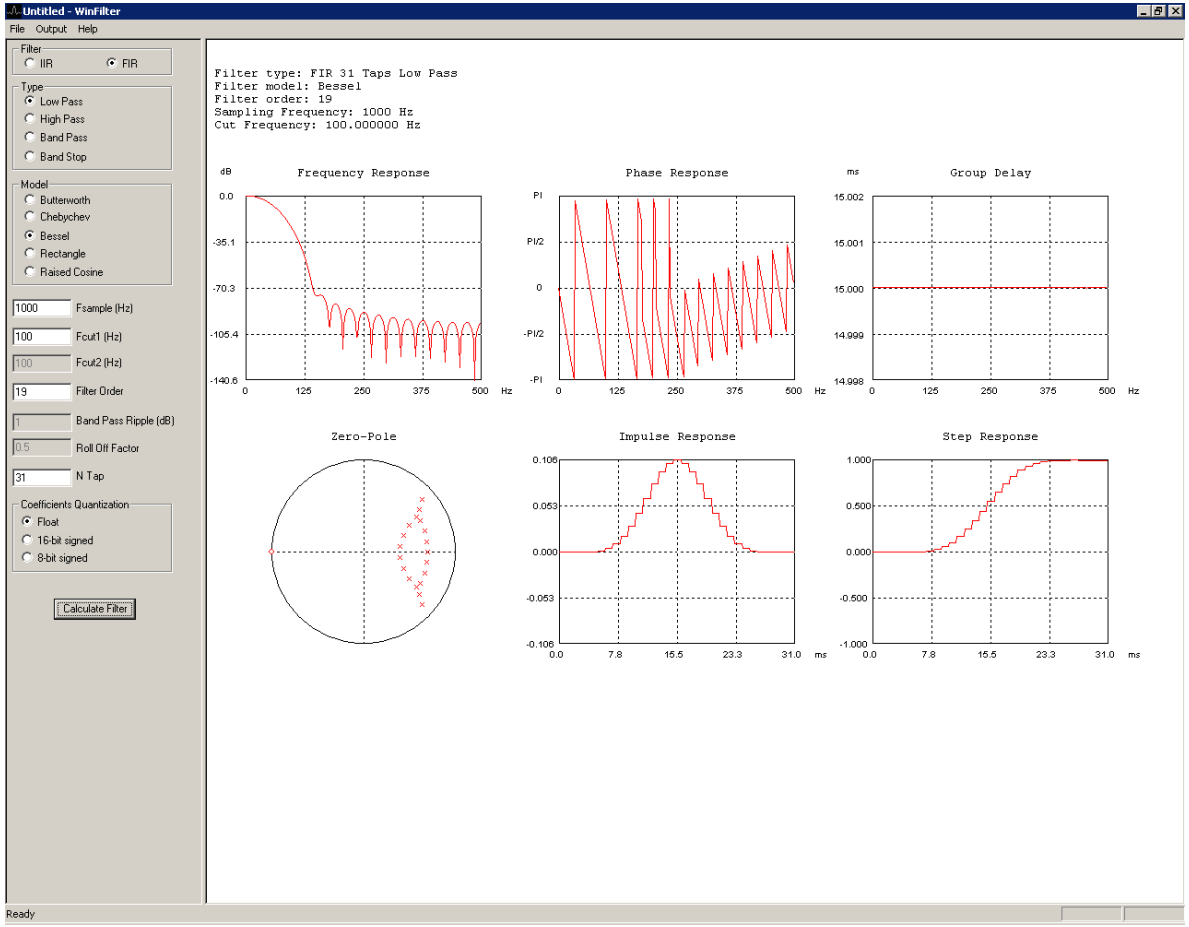


Filters design tools: ScopeFIR

- This design tool is ideal for small projects or DSP enthusiasts.
- Its capabilities are almost the same than fdatool.
- Its price is Us$199, but there is a 60 day trial version.



Filters design tools: ScopeFIR

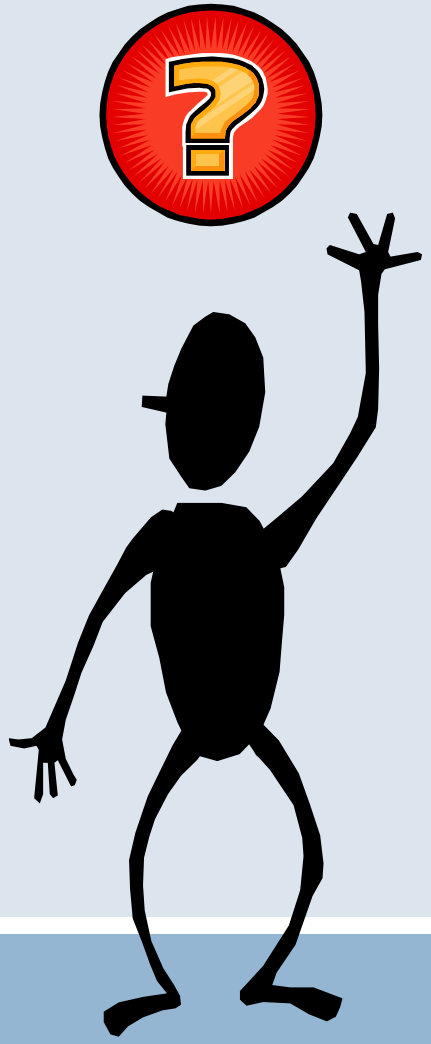


- This design tool is quite limited, but can be useful for students.
- It is completely free.

Recommended bibliography

- RG Lyons, Understanding Digital Signal Processing. Prentice Hall 1997.
  - Ch5 & 6: FIR & IIR filter design.
- SW Smith, The Scientist and Engineer's guide to DSP. California Tech. Pub. 1997.
  - Ch14-21: FIR & IIR filter design.
- EC Ifeachor, BW Jervis. Digital Signal Processing. A Practical approach. Second Edition. Prentice Hall.
  - Ch6: A framework for filter design.
  - Ch7 & 8: FIR & IIR filter design.
  - Ch13: Analysis of wordlength effects in fixed point DSP systems.

NOTE: Many images used in this presentation were extracted from the recommended bibliography.



Questions?

Thank you!