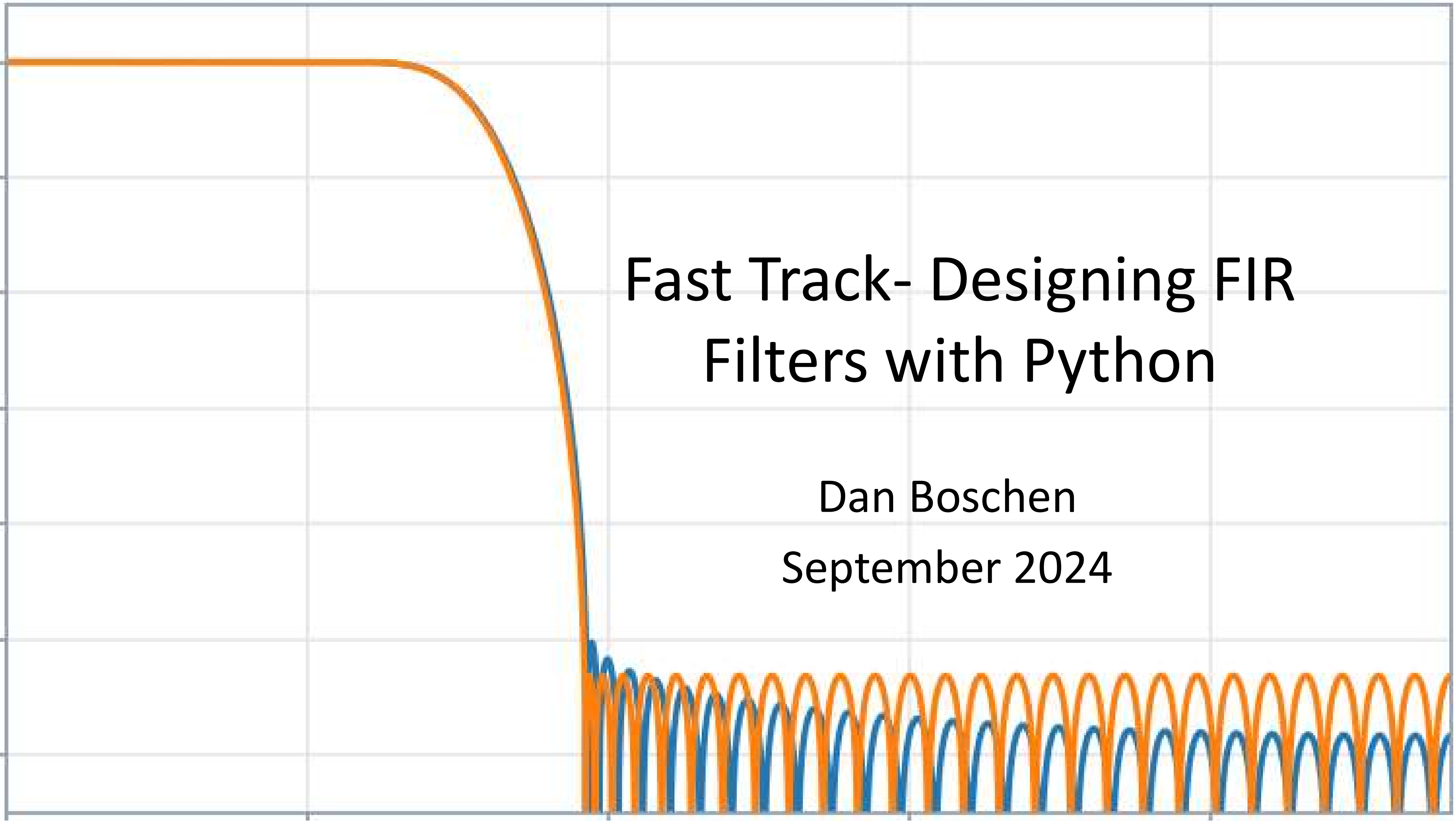
Fast Track- Designing FIR Filters with Python



Dan Boschen

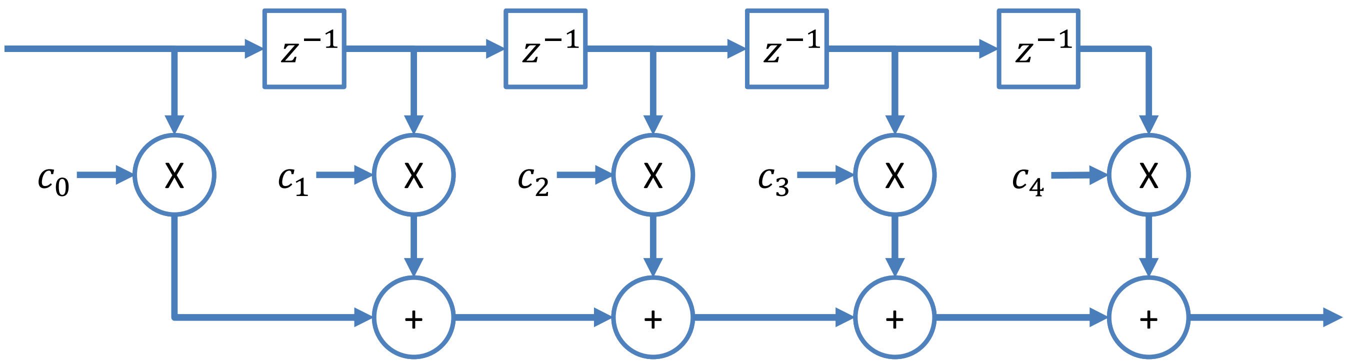
September 2024

Content is derived from courses on DSP and Python taught by Dan Boschen

Copyright © 2024 C. Daniel Boschen
This presentation “Fast Track- Designing FIR Filters with Python” by Dan Boschen is licensed under CC BY-NC-ND 4.0.
To view a copy of this license, visit https://creativecommons.org/licenses/by-nc-nd/4.0/

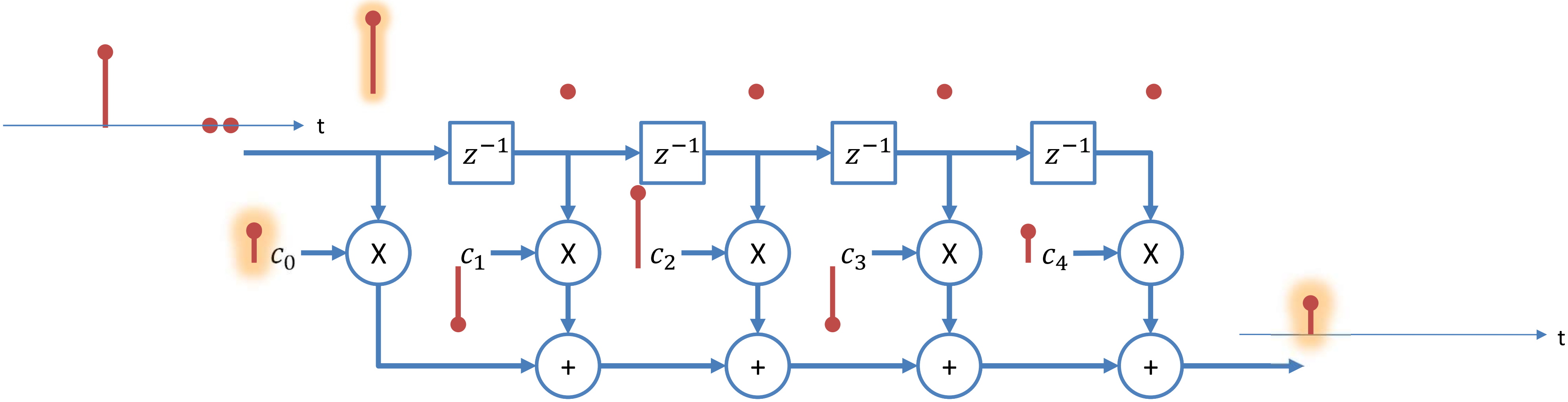
While every precaution has been taken in the preparation of this presentation, the author, publisher, and distribution partners assume no responsibility for any errors or omissions, or any damages resulting from the use of any information contained within it.

FIR Filters



Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

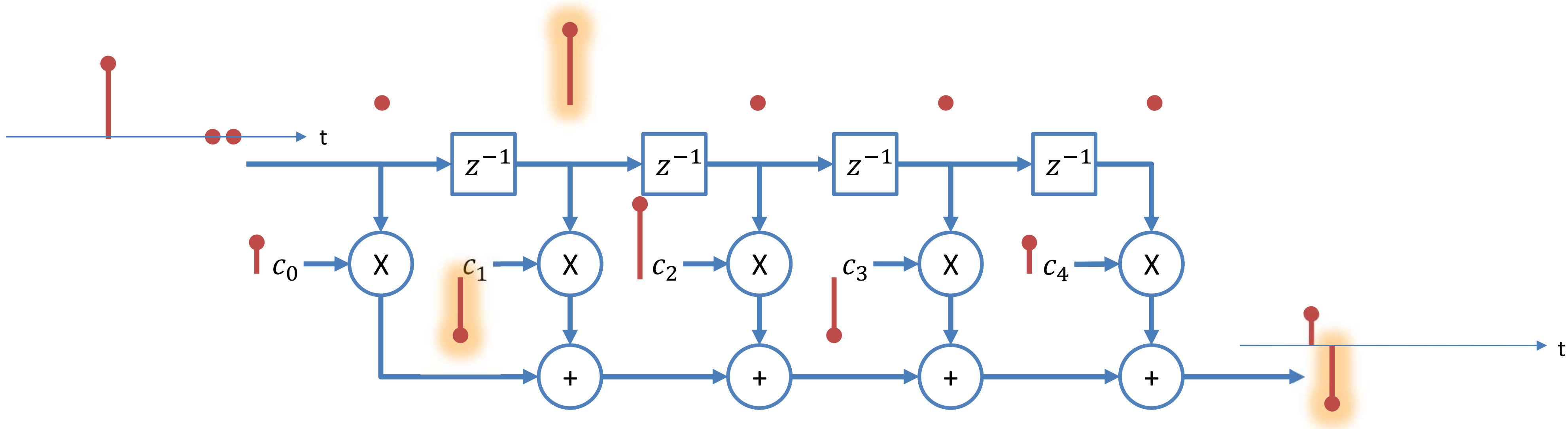
FIR Filters



The Coefficients of the Filter is the **Impulse Response**...

Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

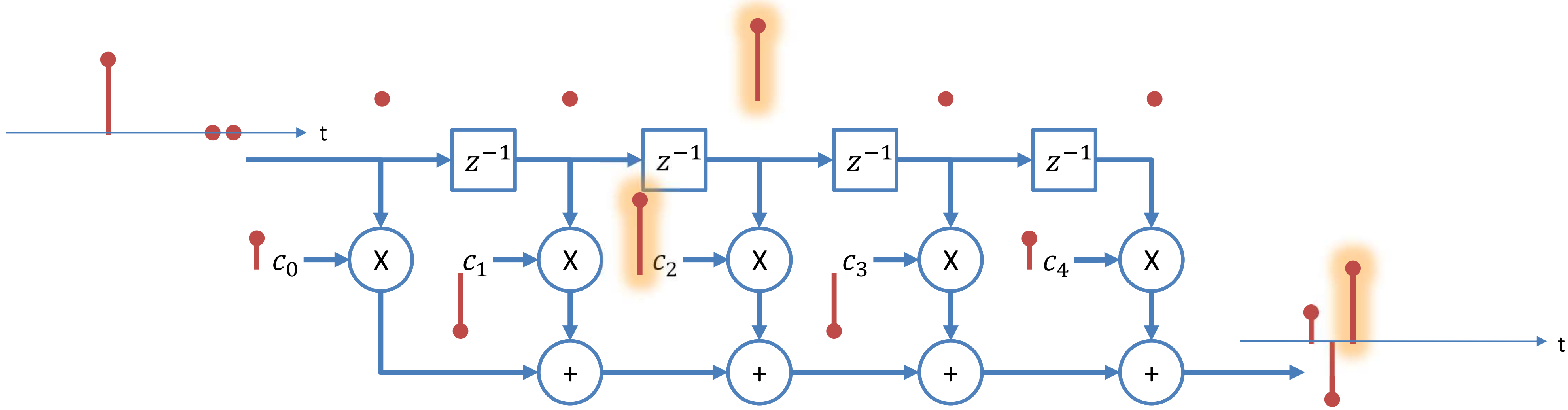
FIR Filters



The Coefficients of the Filter is the **Impulse Response**...

Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

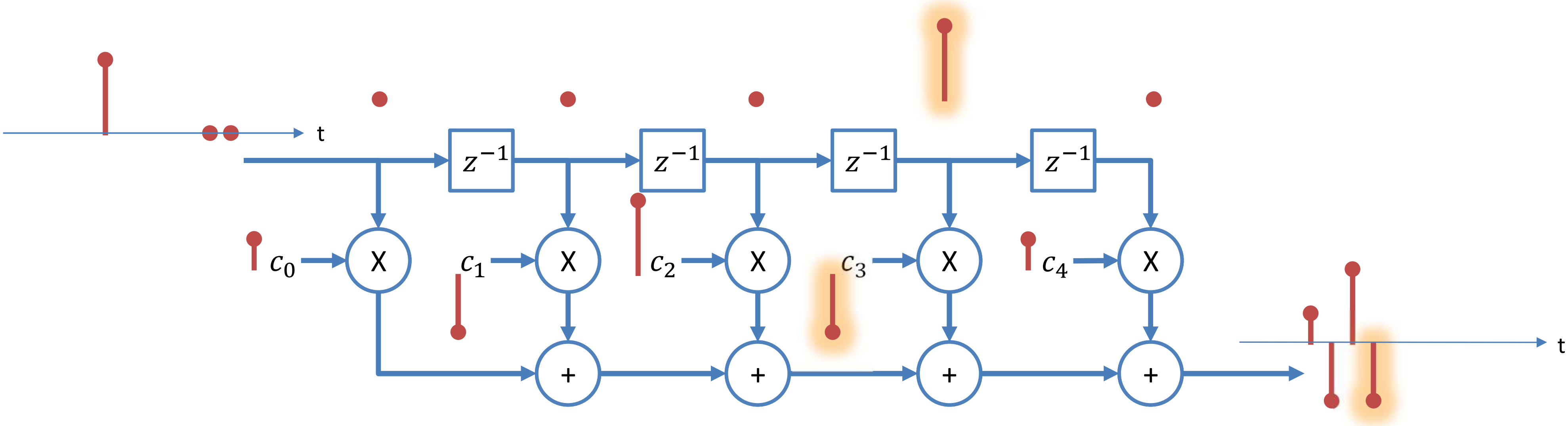
FIR Filters



The Coefficients of the Filter is the **Impulse Response**...

Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

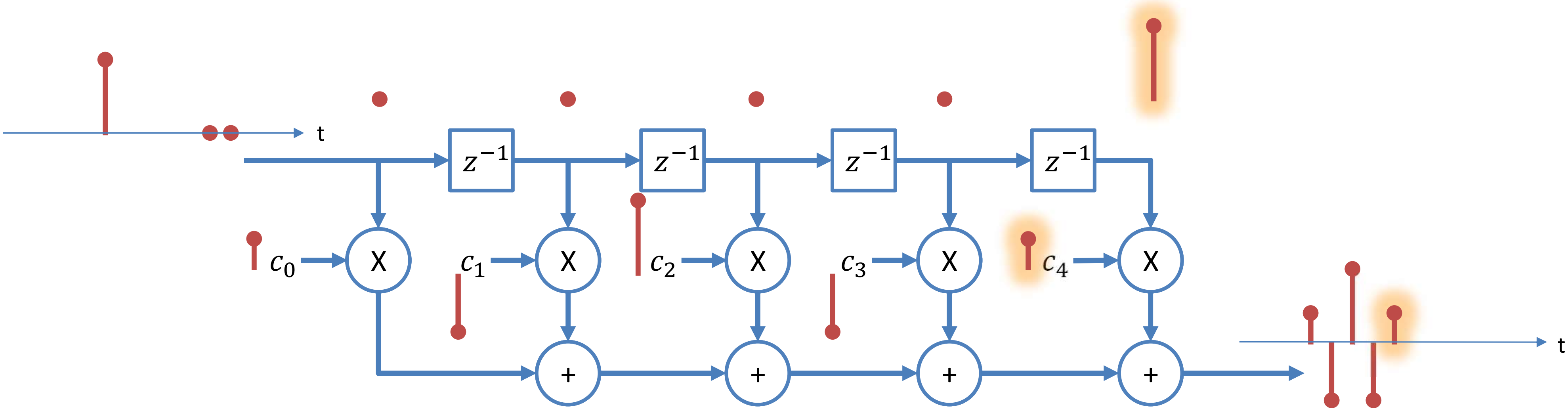
FIR Filters



The Coefficients of the Filter is the **Impulse Response**...

Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

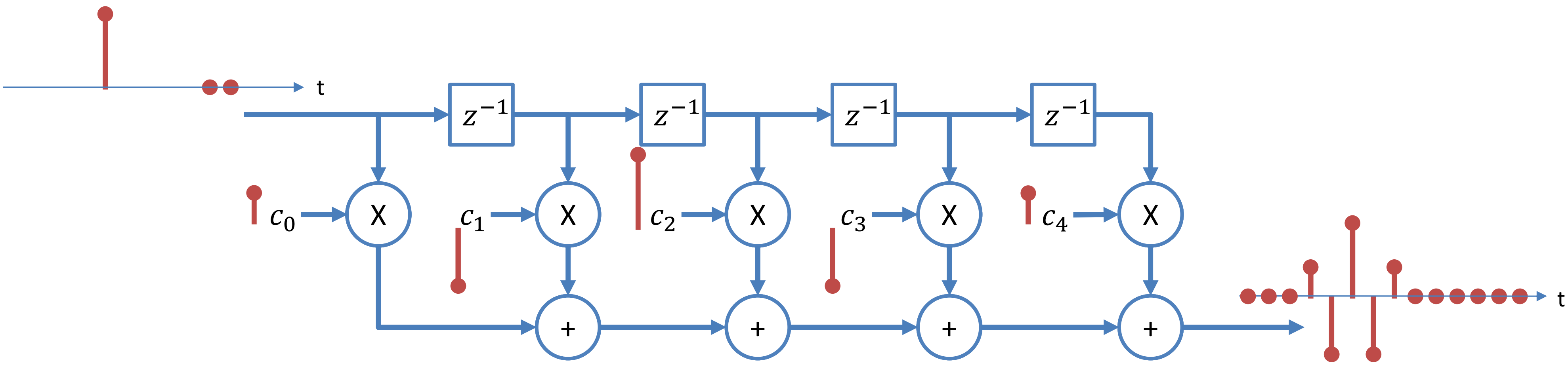
FIR Filters



The Coefficients of the Filter is the **Impulse Response**...

Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

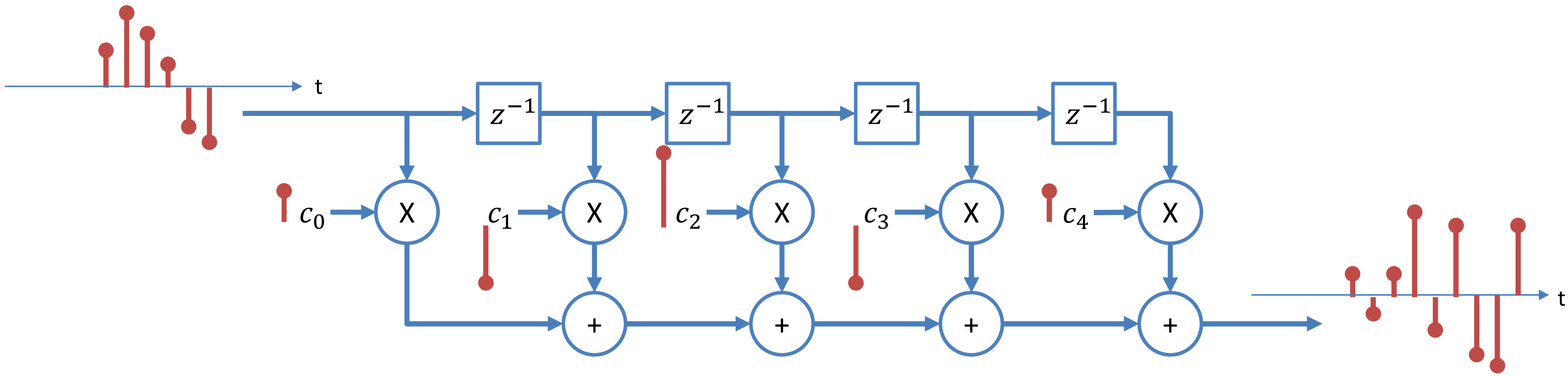
FIR Filters



The Coefficients of the Filter is the **Impulse Response**...

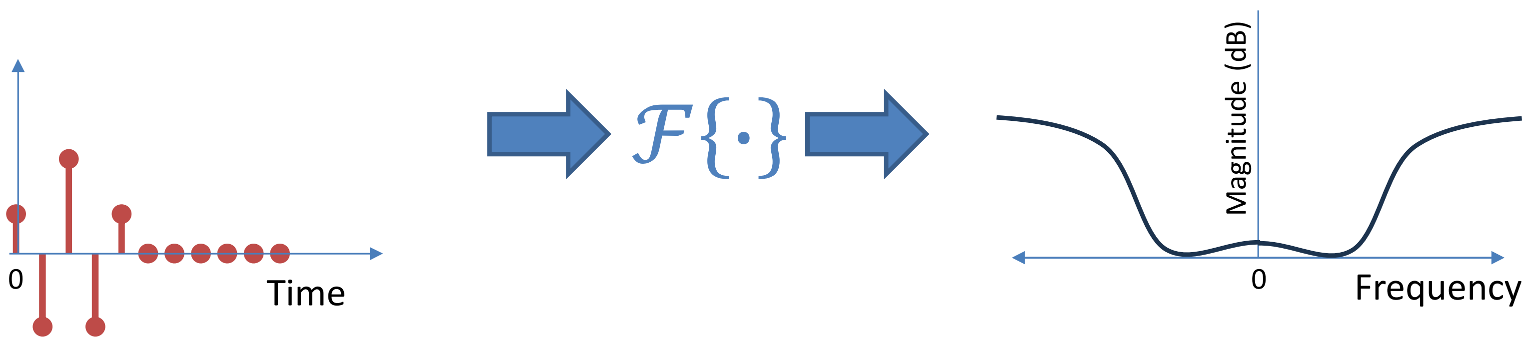
Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

FIR Filters



The output is the **convolution** of the input with the coefficients...

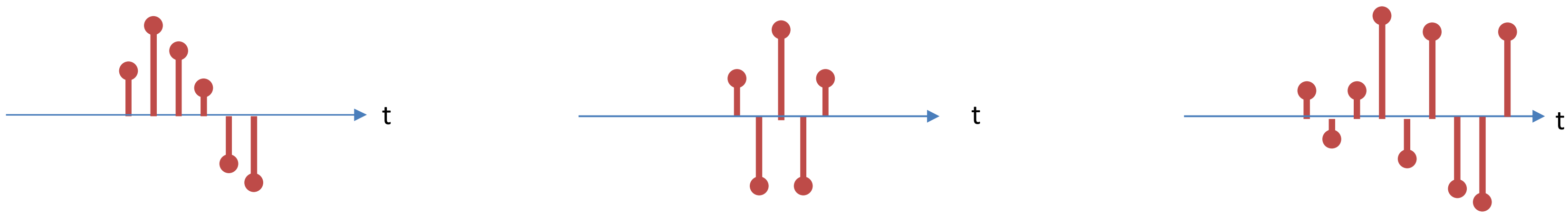
Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0



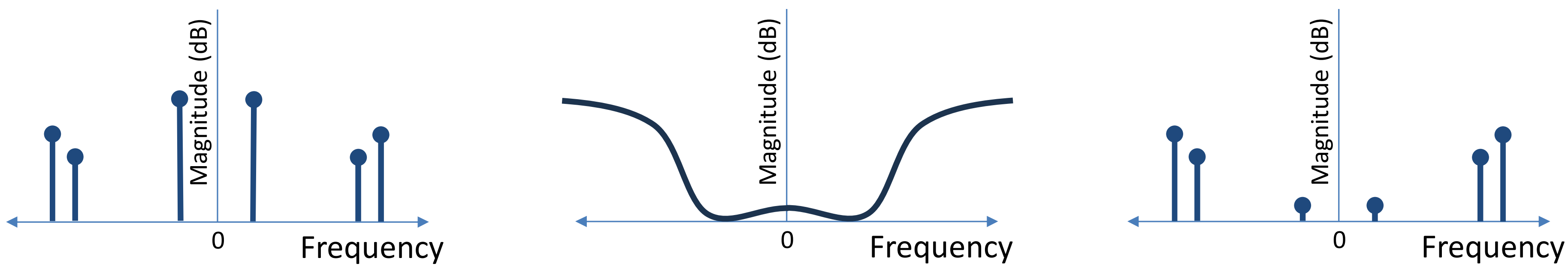
The Fourier Transform of the Impulse Response is the **Frequency Response**

Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

Convolution in the time domain …

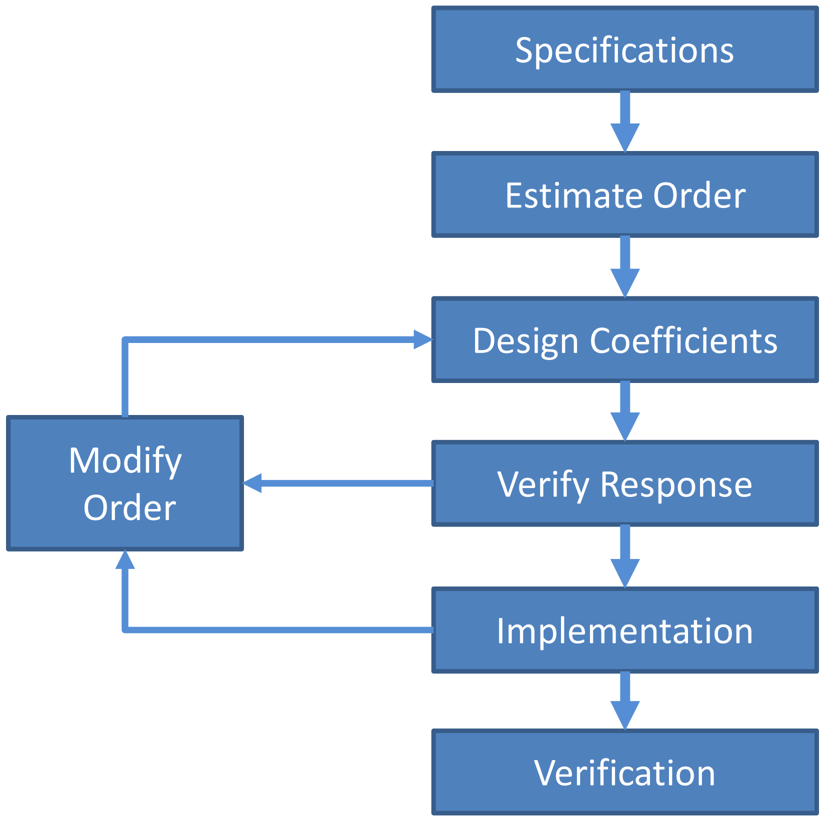


… is **Multiplication** in the frequency domain



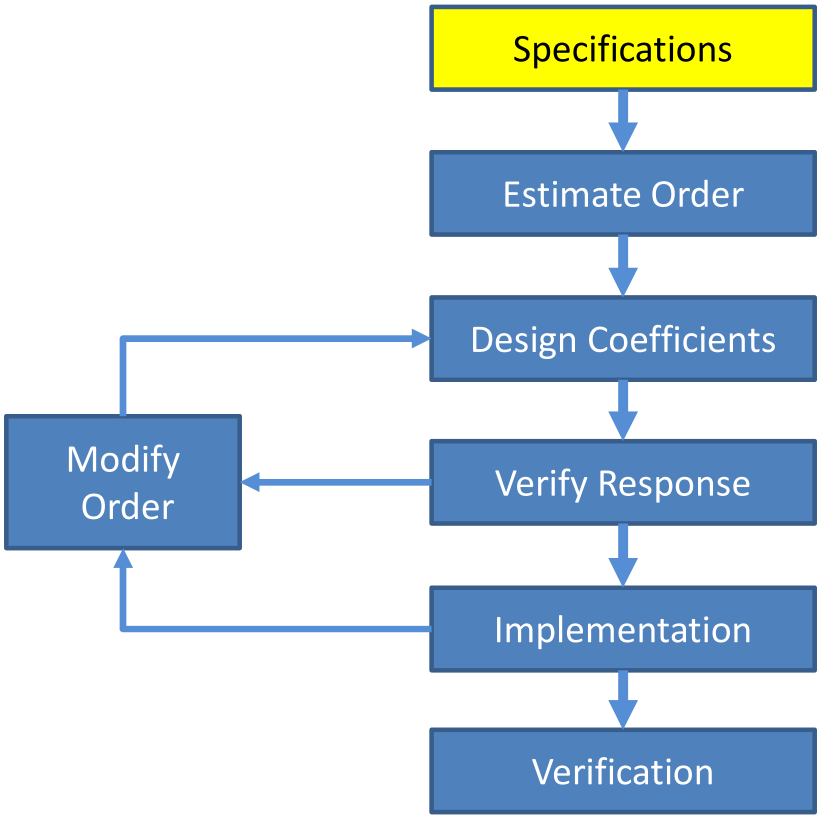
Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

FIR Design Flow



Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

FIR Design Flow



Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

Filter Specification

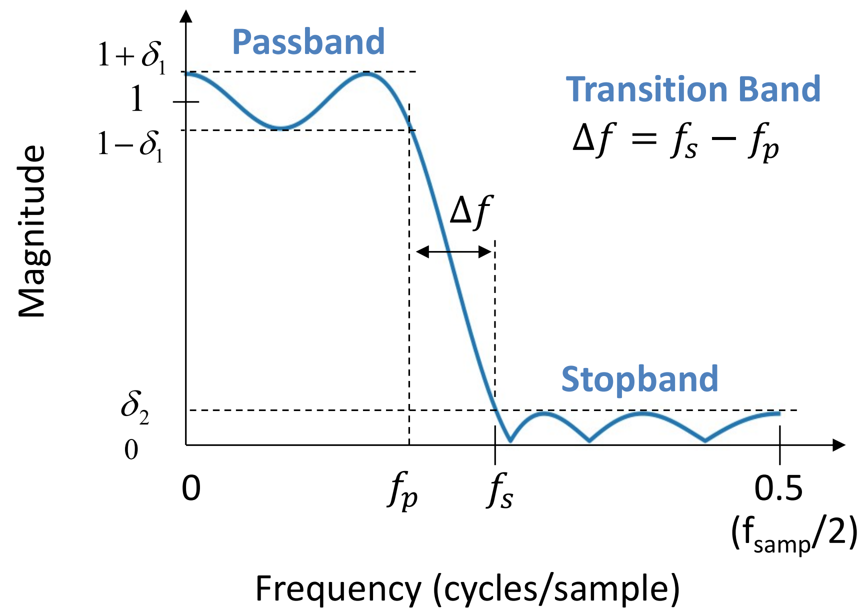
f_p: **Passband edge frequency** (cycles/sample)

δ_1: **Peak Passband Ripple**, in dB: $-20log_{10}(1-\delta_1)$

f_s: **Stopband edge frequency** (cycles/sample)

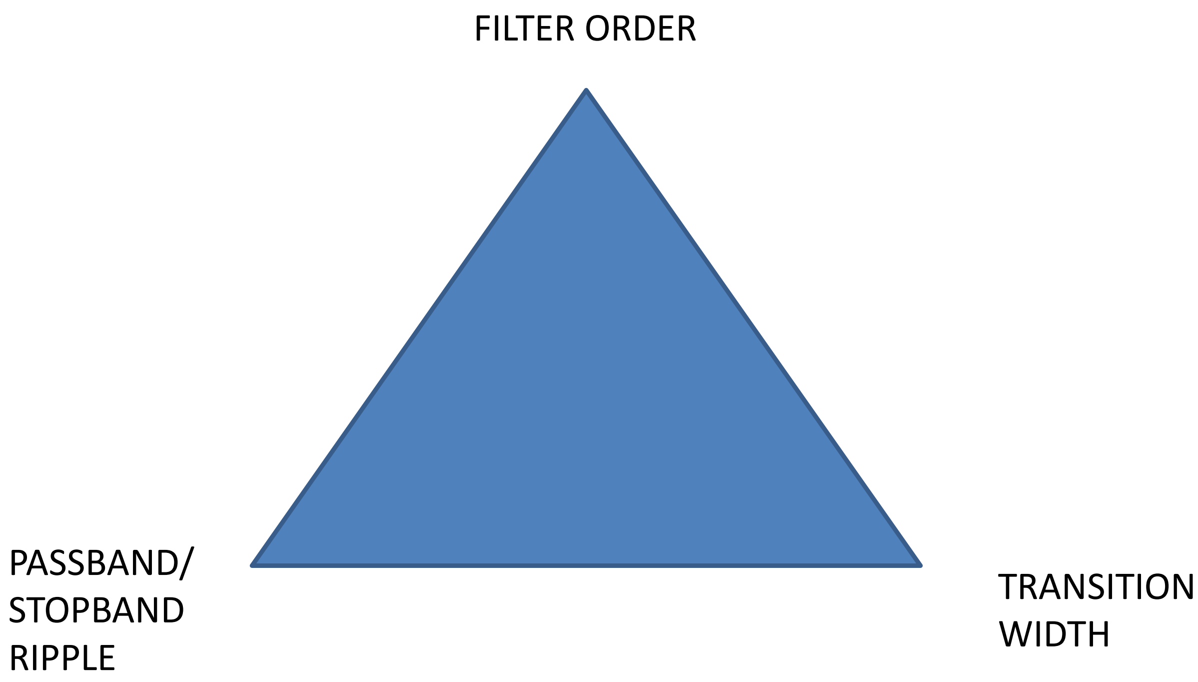
δ_2: **Peak Stopband Ripple**, in dB (rejection): $20log_{10}(\delta_2)$

Filter order: Total time delay or processing constraint



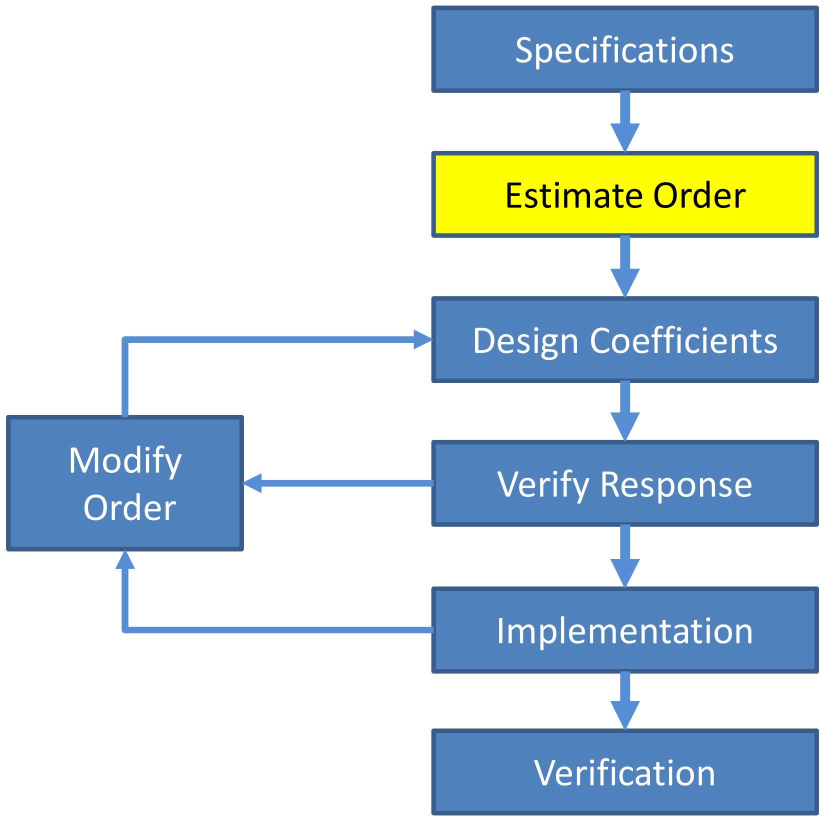
Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

Trade Space



Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

FIR Design Flow



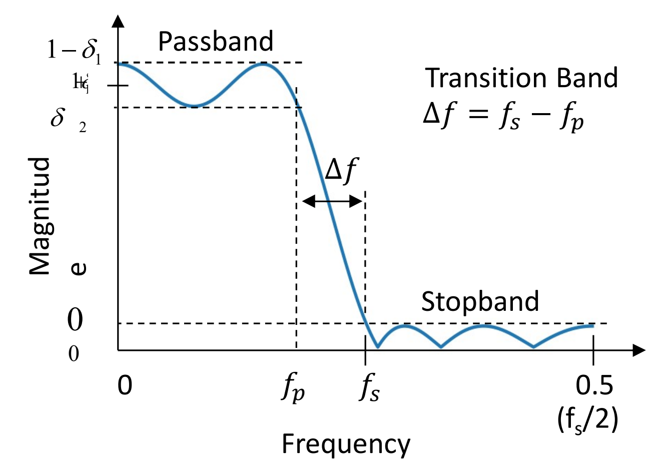
Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

Estimate Number of Coefficients

`scipy.signal.kaiserord`

Kaiser's Estimators	
Parks-McClellan Filters	Kaiser Window Filters
$\dfrac{-10log_{10}(\delta_1\delta_2) - 13}{14.6(\Delta f)} + 1$	$\dfrac{A - 8}{14.4(\Delta f)} + 1$

fred harris Estimator
$\dfrac{A}{22(\Delta f)}$

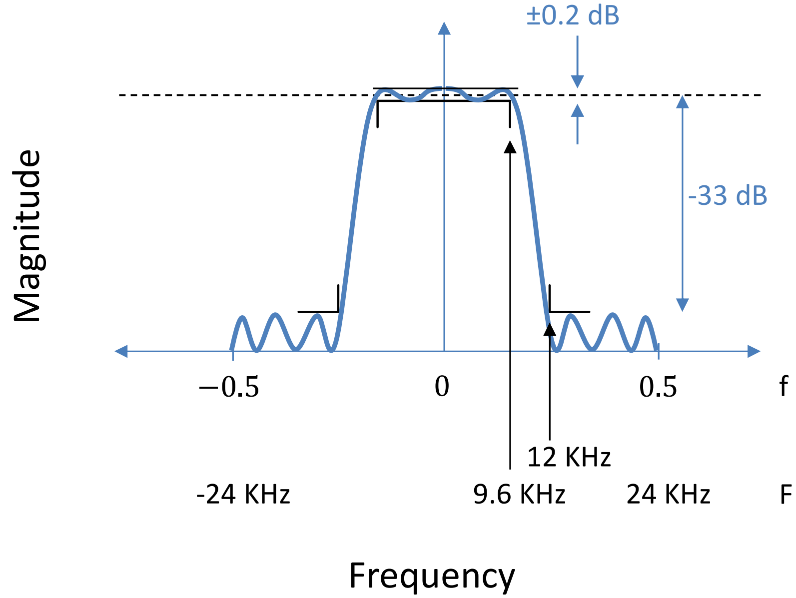


Where A is attenuation in dB as a positive number:
$$A = -20log_{10}(\delta_2)$$

Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

Example

<u>Low Pass Filter</u>
Passband ripple ±0.2 dB
Stopband rejection: -33 dB
Sampling rate: 48 KHz
Passband edge: 9.6 KHz
Stopband edge: 12 KHz



Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

Example

<u>Low Pass Filter</u>

Passband ripple ±0.2 dB

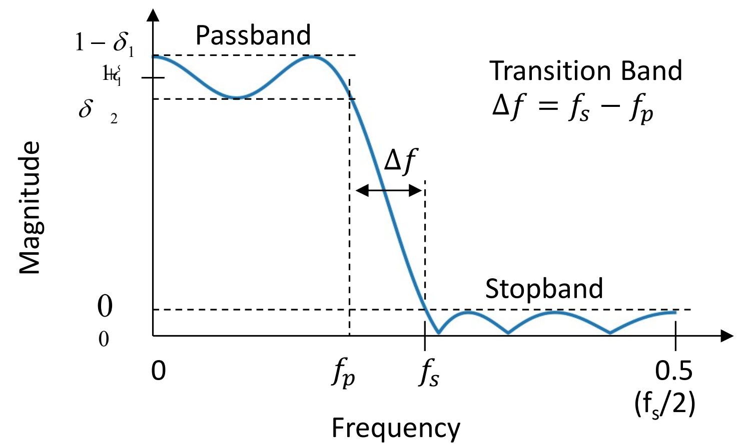
Stopband rejection: -33 dB

Sampling rate: 48 KHz

Passband edge: 9.6 KHz

Stopband edge: 12 KHz

$$\delta_1 = 10^{0.2/20} - 1 = 0.023$$
$$\delta_2 = 10^{-33/20} = 0.022$$
$$f_p = \frac{9600}{48000} = 0.2$$
$$f_s = \frac{12000}{48000} = 0.25$$

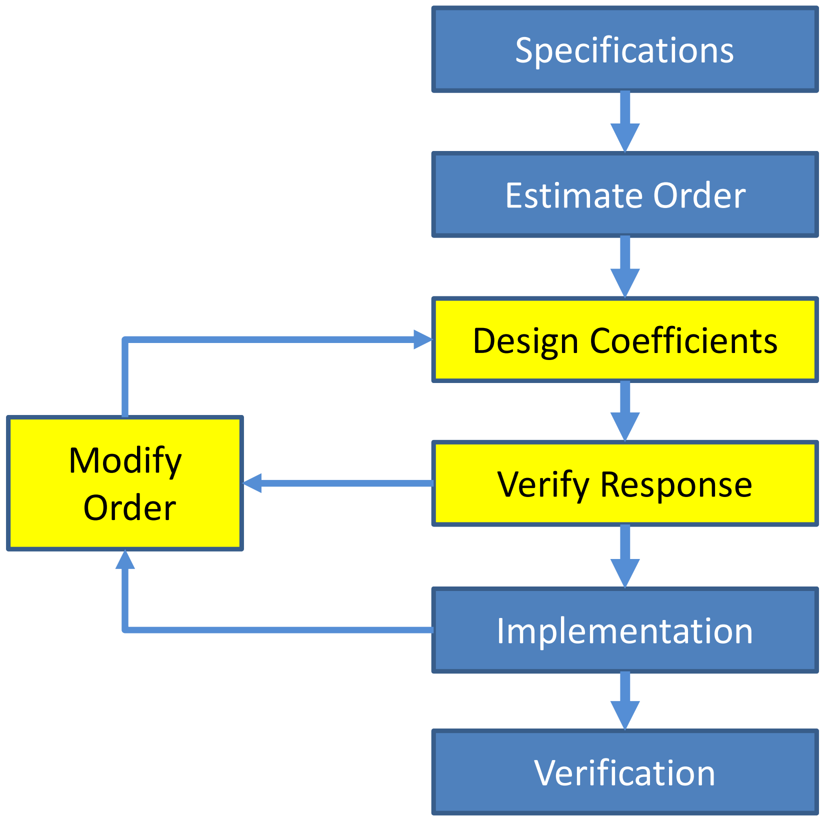


Kaiser: $$N \cong \frac{-10log_{10}(\delta_1\delta_2) - 13}{14.6(\Delta f)} + 1 = \frac{-10log_{10}(0.022 \cdot 0.023) - 13}{14.6 \cdot 0.05} = 28.3$$

fred harris: $$N \cong \frac{A}{22(\Delta f)} = \frac{33}{22 \cdot 0.05} = 30$$

Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

FIR Design Flow



Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

Direct FIR Realizations

Window method

Frequency Sampling

Equiripple (Parks-McClellan or Remez)

Least-Squares

Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

Direct FIR Realizations

Window method

Frequency Sampling

Equiripple (Parks-McClellan or Remez)

Least-Squares

Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

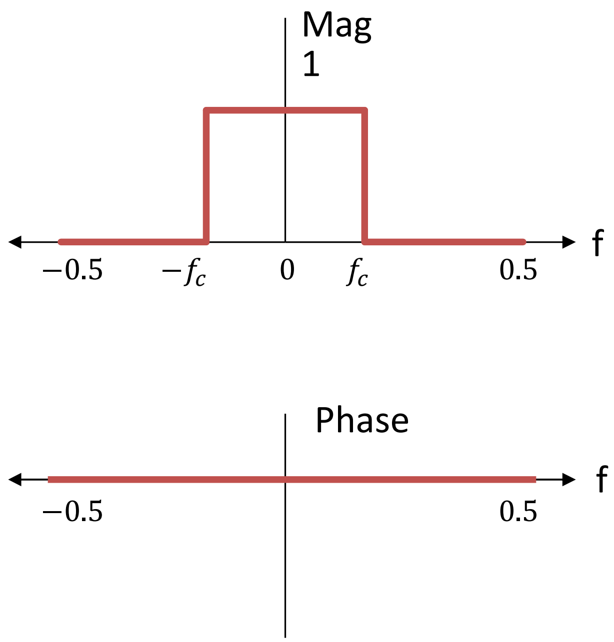
Truncate and Window

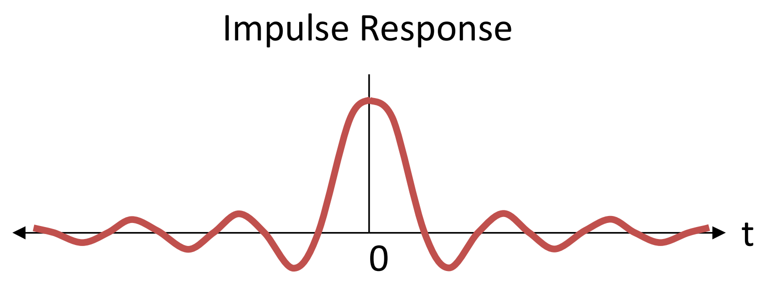
- Simple
- Non-optimal
- Typical windows
  - Bartlett, Modified Bartlett-Hanning (“Barthann”), Blackman, Blackman-harris, Bohman, Chebyshev, Cosine, DPSS, Flat Top, Gaussian, Hamming, Hann, Kaiser, Nutall, Parzen (de la Valle-Poussin), Boxcar, Taylor, Cosine-tapered (Tukey), Triangular
- All of these windows are available in Python!
- https://docs.scipy.org/doc/scipy/reference/signal.windows.html

Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

Ideal Lowpass Filter

$H(f) = 1, 0 \leq |f| \leq f_c$
$H(f) = 0, |f| > f_c$

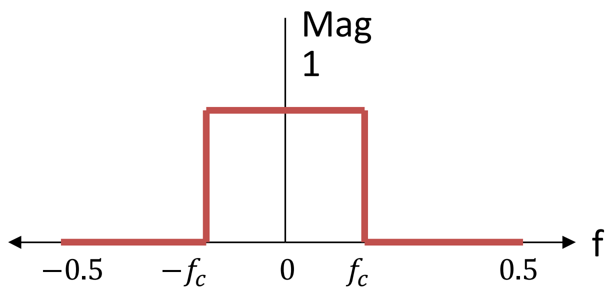


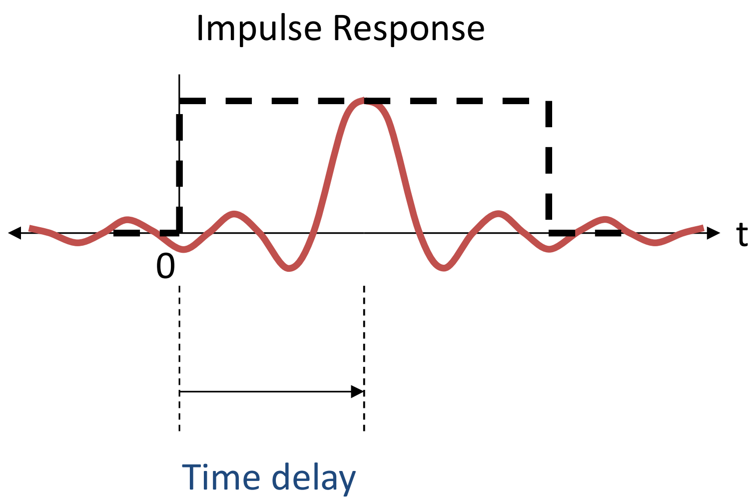


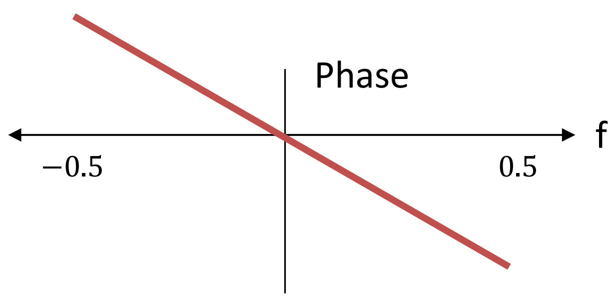
Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

Ideal Lowpass Filter

$H(f) = 1, 0 \le |f| \le f_c$
$H(f) = 0, |f| > f_c$



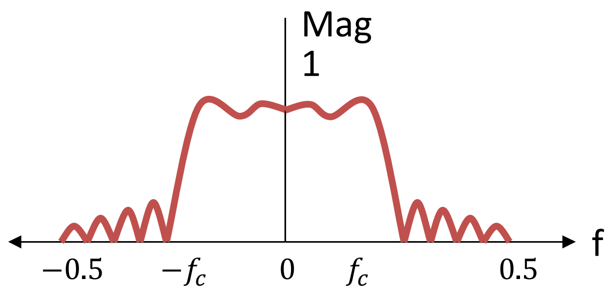


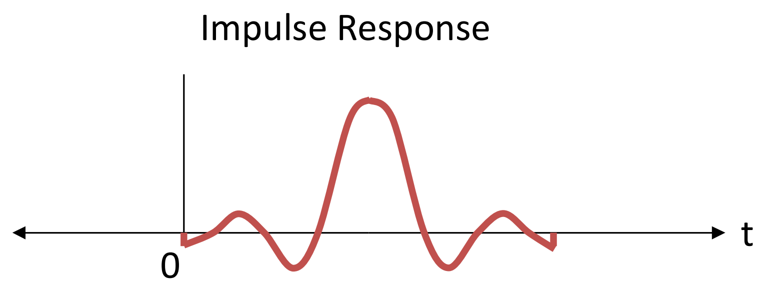


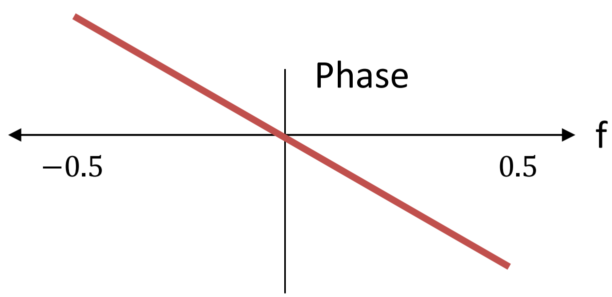
Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

Rectangular Window

$H(f) = 1, 0 \leq |f| \leq f_c$
$H(f) = 0, |f| > f_c$

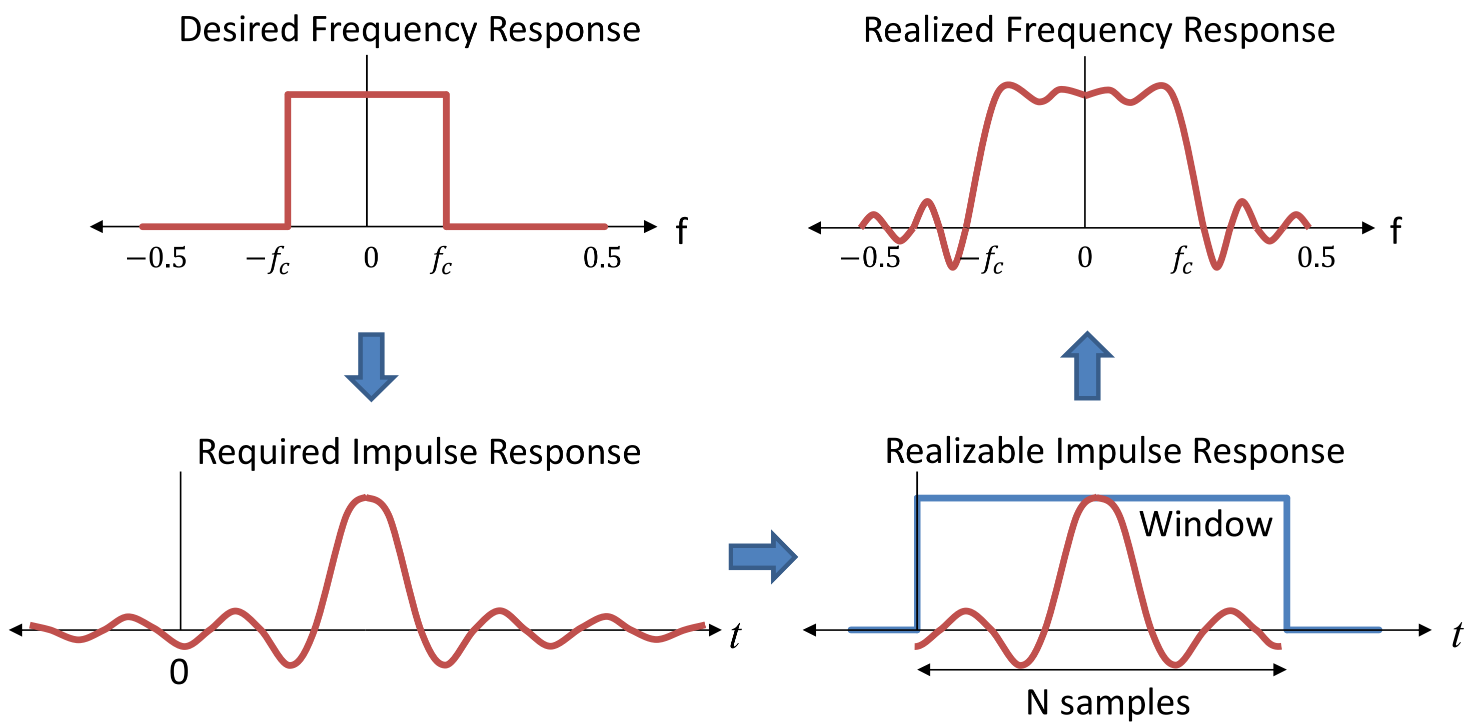






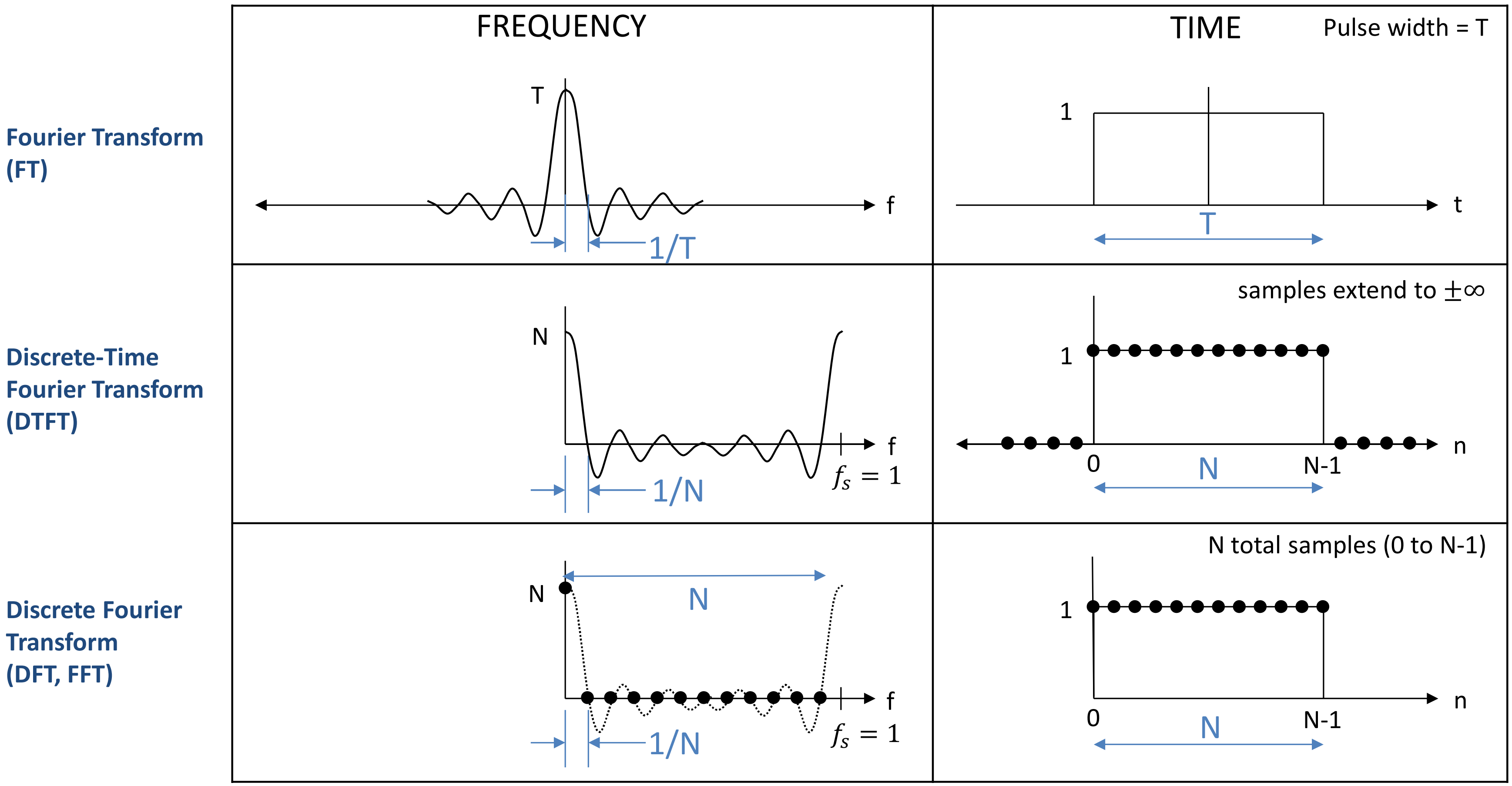
Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

Rectangular Window



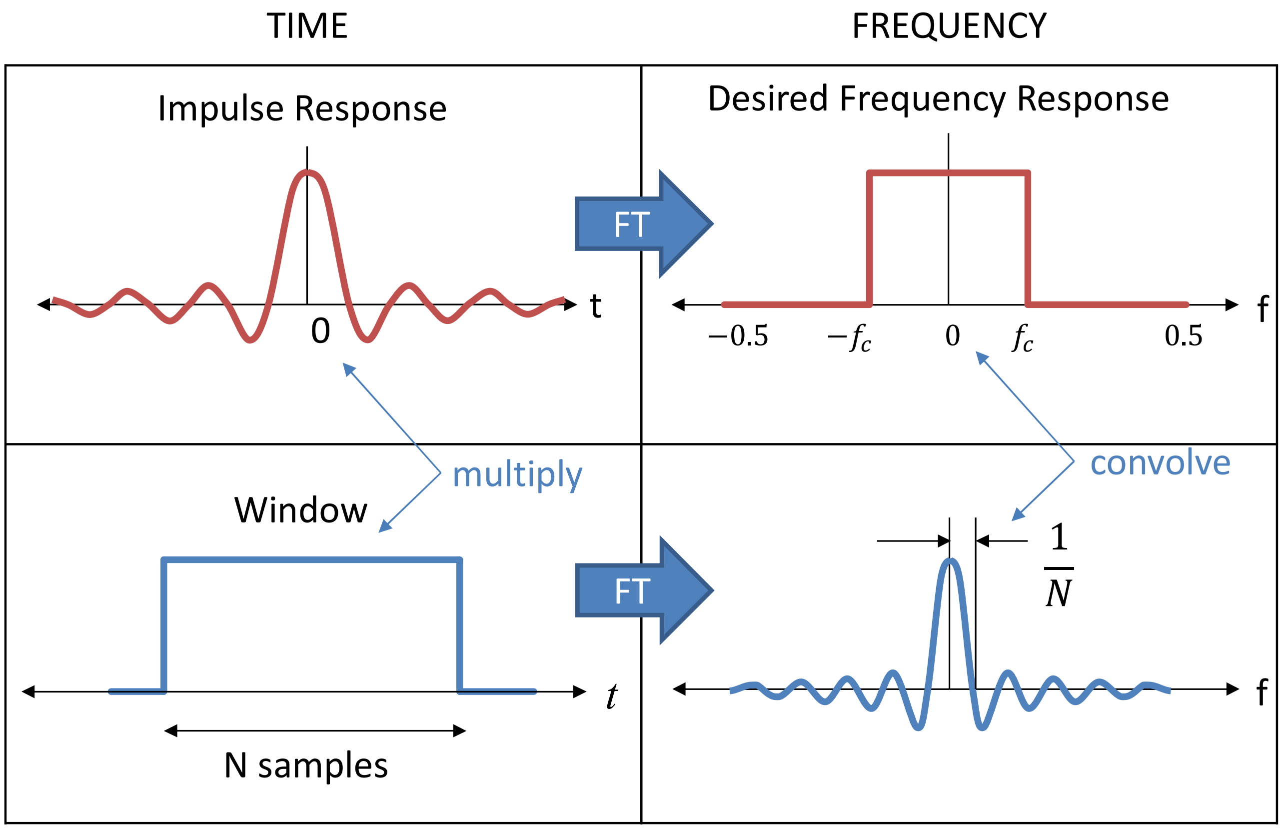
Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

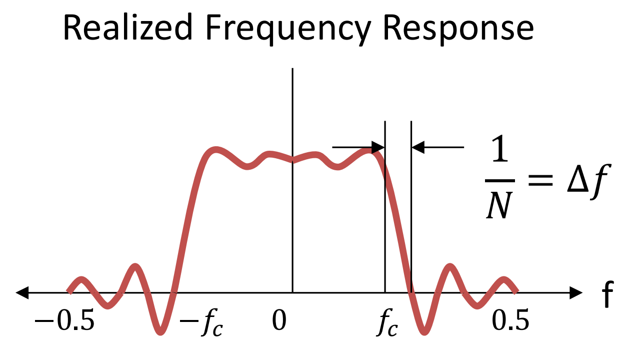
Transform Review



Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

Rectangular Window



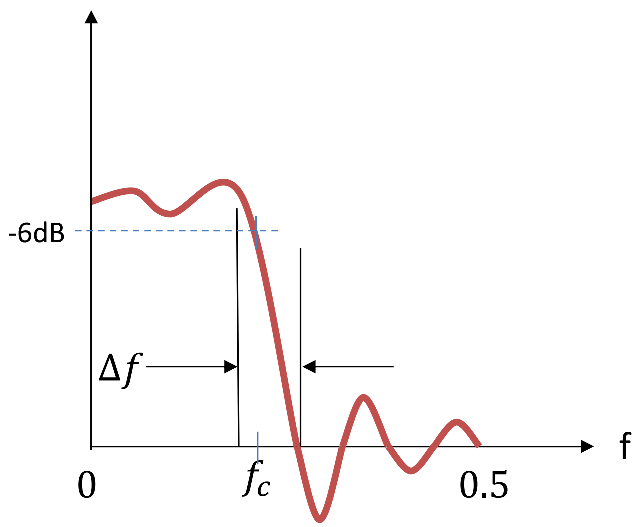


$$N = \frac{1}{\Delta f}$$

Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

Rectangular Window

$N = \frac{1}{\Delta f}$ coefficients required to complete transition band



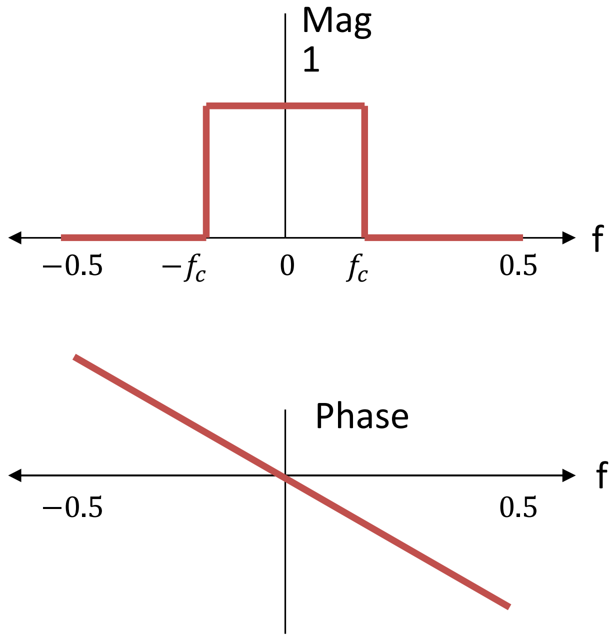
Kaiser's Estimators	
Parks-McLellan Filters	Kaiser Window Filters
$\frac{-10log_{10}(\delta_1\delta_2)-13}{14.6(\Delta f)}+1$	$\frac{A-8}{14.4(\Delta f)}+1$

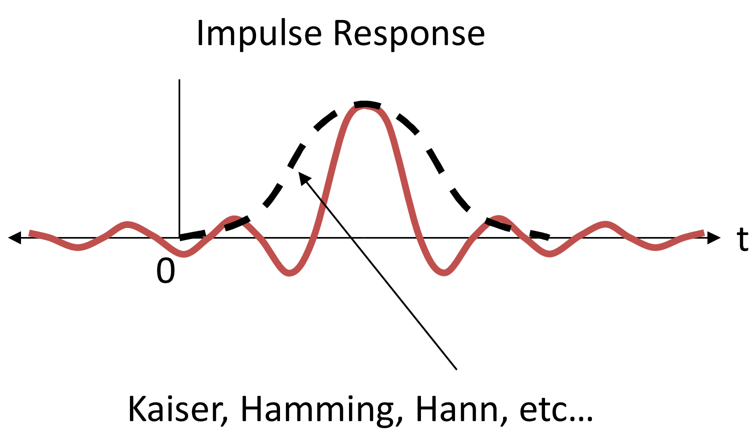
fred harris Estimator
$N \cong \frac{A}{22(\Delta f)}$

Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

Alternate Window

$H(f) = 1, 0 \leq |f| \leq f_c$
$H(f) = 0, |f| > f_c$

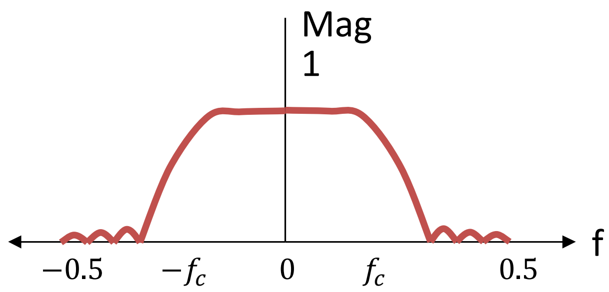


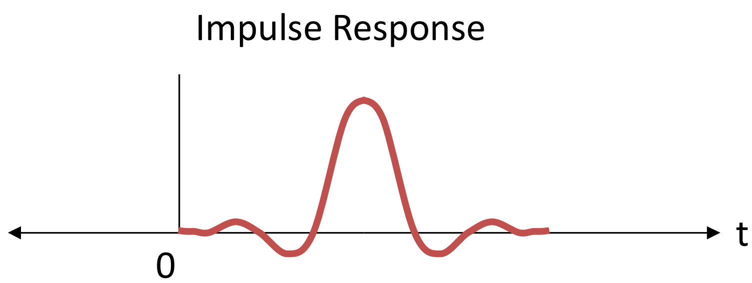


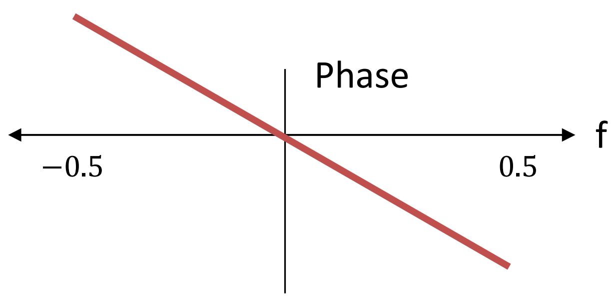
Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

Alternate Window

$H(f) = 1, 0 \leq |f| \leq f_c$
$H(f) = 0, |f| > f_c$



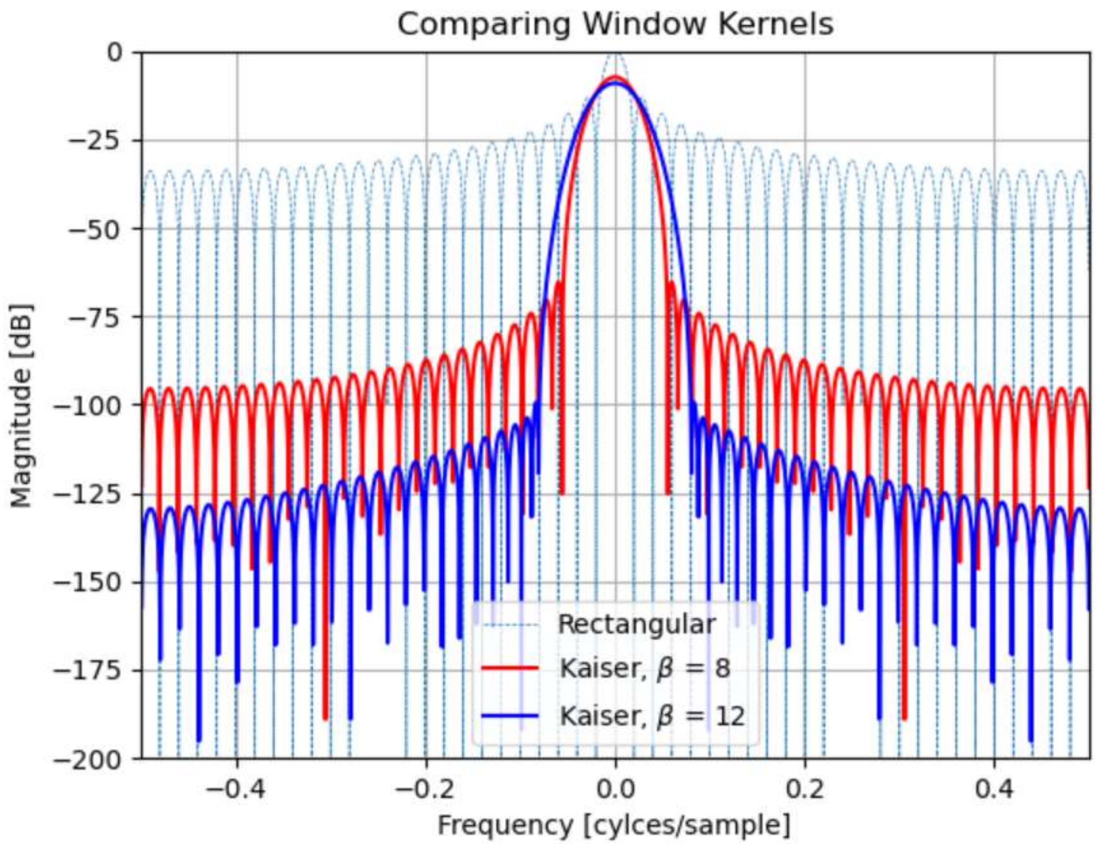




Reduced passband
and stopband ripple!
…But wider transition band
for a given length N

Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

DTFT of the Kaiser Window



Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

Window Summary

Establish desired frequency response

Determine impulse response (using Inverse Fourier Transform)

Sample, truncate and window

Estimate number of taps and Kaiser beta:
`scipy.signal.kaiserord`

Create filter coefficients
`scipy.signal.firwin` OR
`scipy.signal.firwin2`

Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

Direct FIR Realizations

Window method

Frequency Sampling

Equiripple (Parks-McClellan or Remez)

Least-Squares

Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

Frequency Sampling

<u>Very</u> simple

Coefficients are the Inverse DFT of the sampled frequency response.

(In contrast to Inverse FT for Windowing method).

Sub-optimal

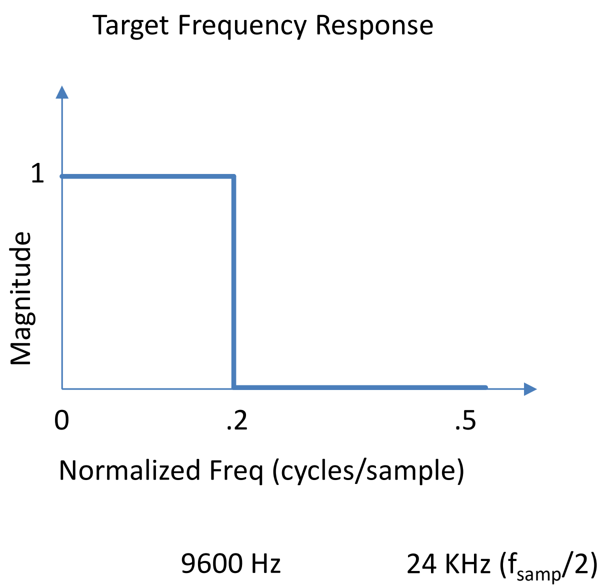
Longer filter length required over optimized approaches.

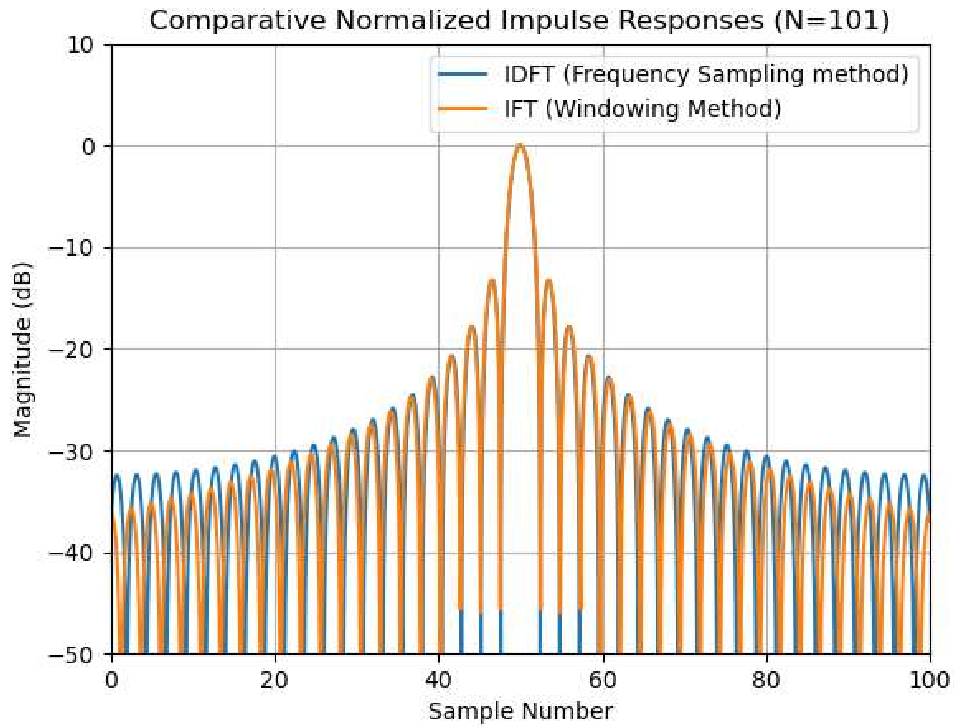
Will be exact at frequency sample locations.

Larger ripple in between.

Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

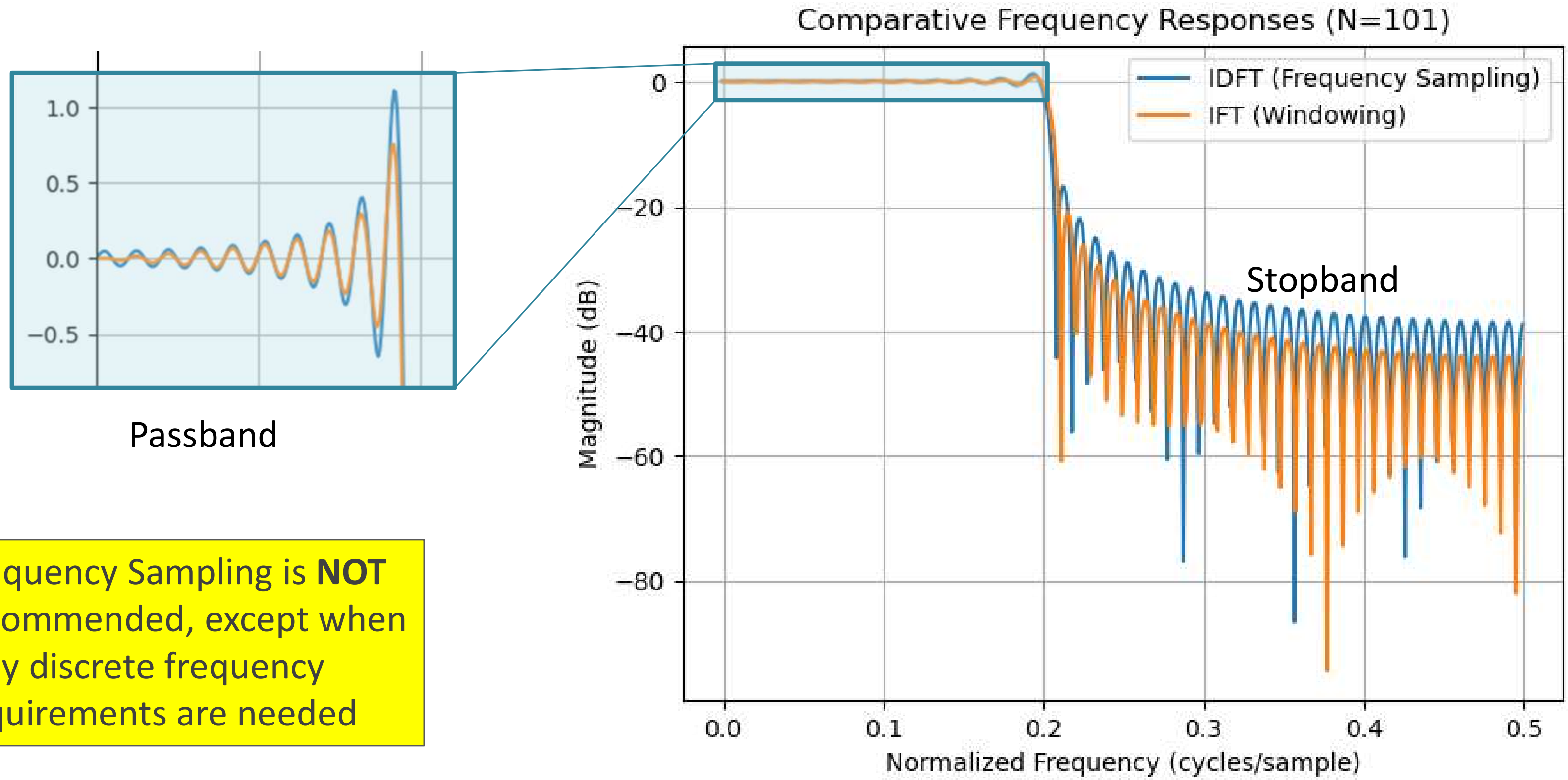
Frequency Sampling Example





Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

Frequency Sampling



Frequency Sampling is **NOT** recommended, except when only discrete frequency requirements are needed

Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

Direct FIR Realizations

Window method

Frequency Sampling

Equiripple (Parks-McClellan or Remez)

Least-Squares

Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

Optimized Algorithms

Equiripple (Parks-McClellan, Remez)

“best” in the mini-max sense: minimize the maximum error

Ideally suited when a specific tolerance must be met

Smallest peak error in passband and stopband

Least Squares

“best” in the least squares sense: minimize the least square error

Will have greater excursions from target than Equiripple, but lower rms error overall.

Smallest stopband energy and rms passband error

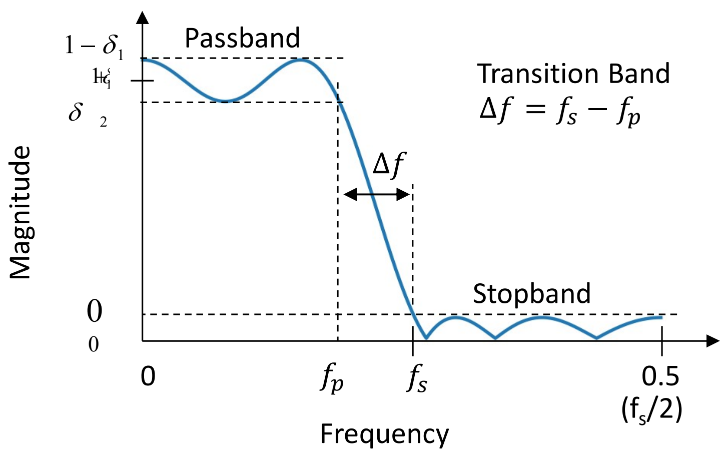
Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

Design Tools for Optimized Algorithms

	Python scipy.signal	MATLAB	Octave
Equiripple	remez	firpm	remez
Least Squares	firls	firls	firls

Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

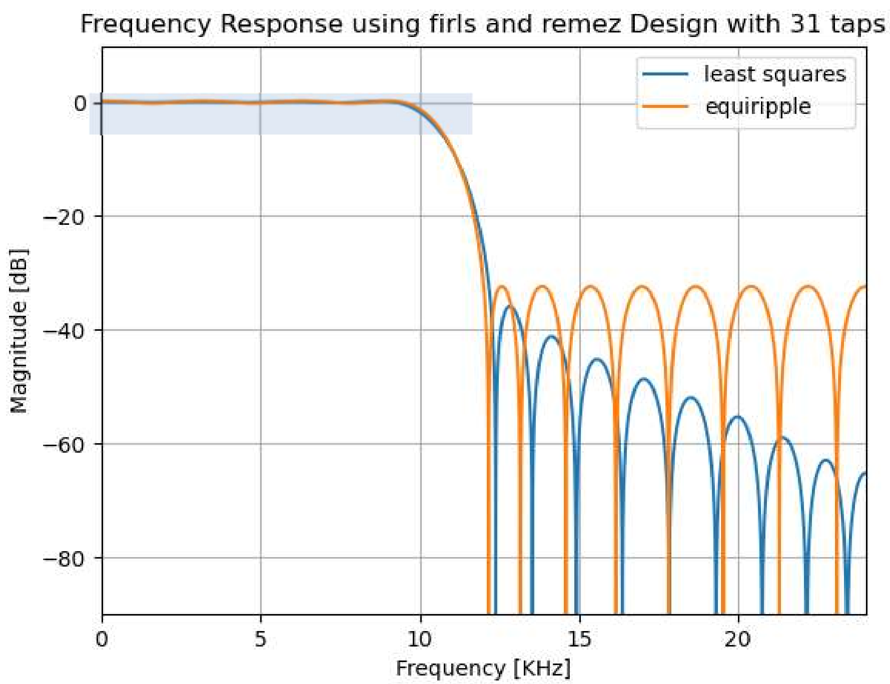
Design Tools for Optimized Algorithms



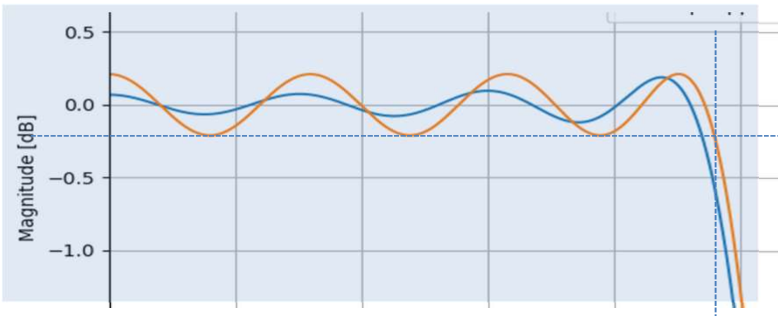

```
# bands (nyq = 1 for firls, nyq = 0.5 for remez)
ls_bands = [0, 2*pass_edge, 2*stop_edge, nyq]
remez_bands = [0, pass_edge, stop_edge, nyq]

ls_coeff = sig.firls(n_taps, ls_bands, [1,1,0,0])
remez_coeff = sig.remez(n_taps, remez_bands, [1, 0])
```


Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

Example



Equiripple vs Least-Squared



Passband

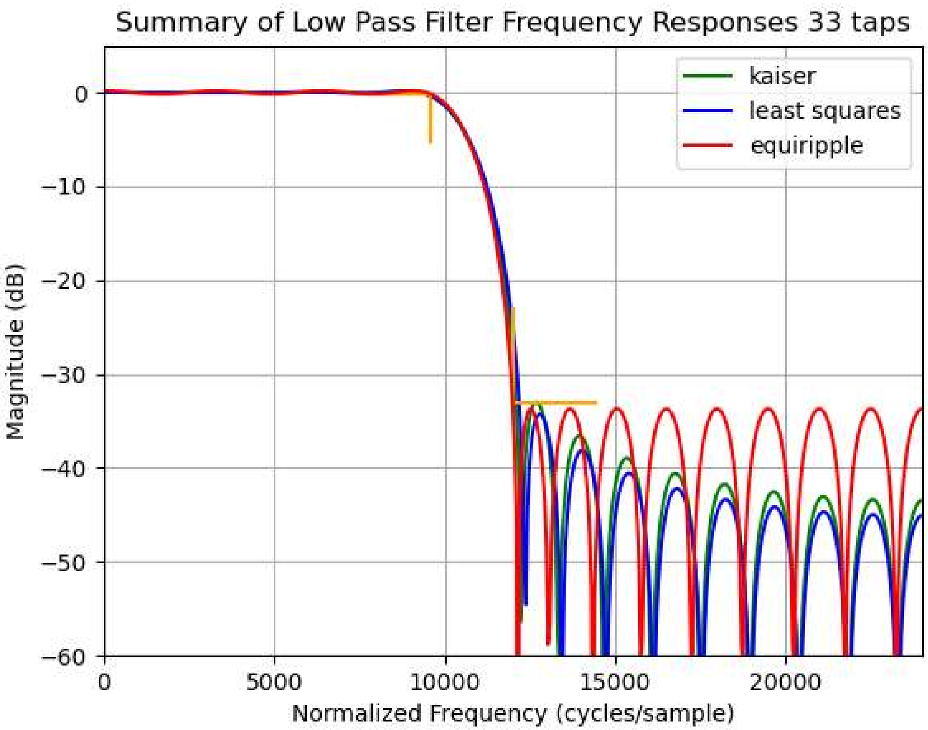
9.6KHz

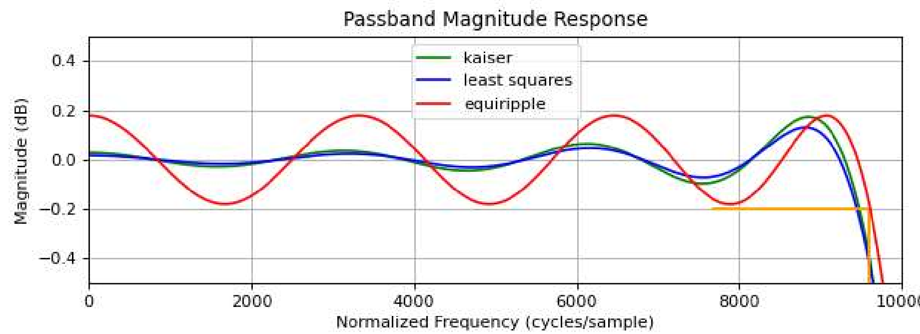
9.6 KHz Pass
12 KHz Stop
Fs=48 KHz
N = 31

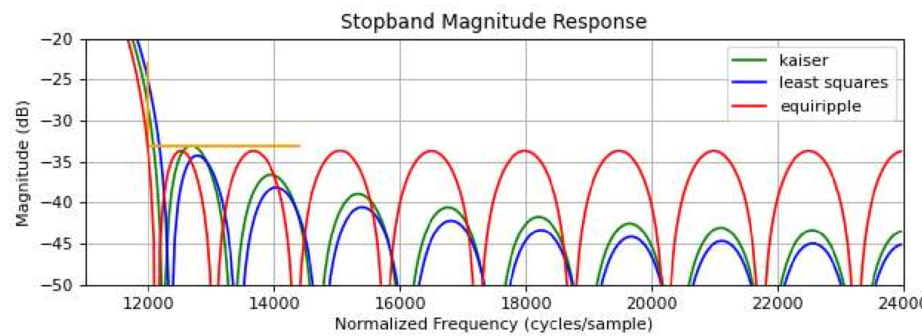
```
ls_bands = [0, 9.6/2.4, 12/24, 1]
remez_bands = [0, 9.6/48, 12/48, 0.5]
```


Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

Performance Summary of Design Methods

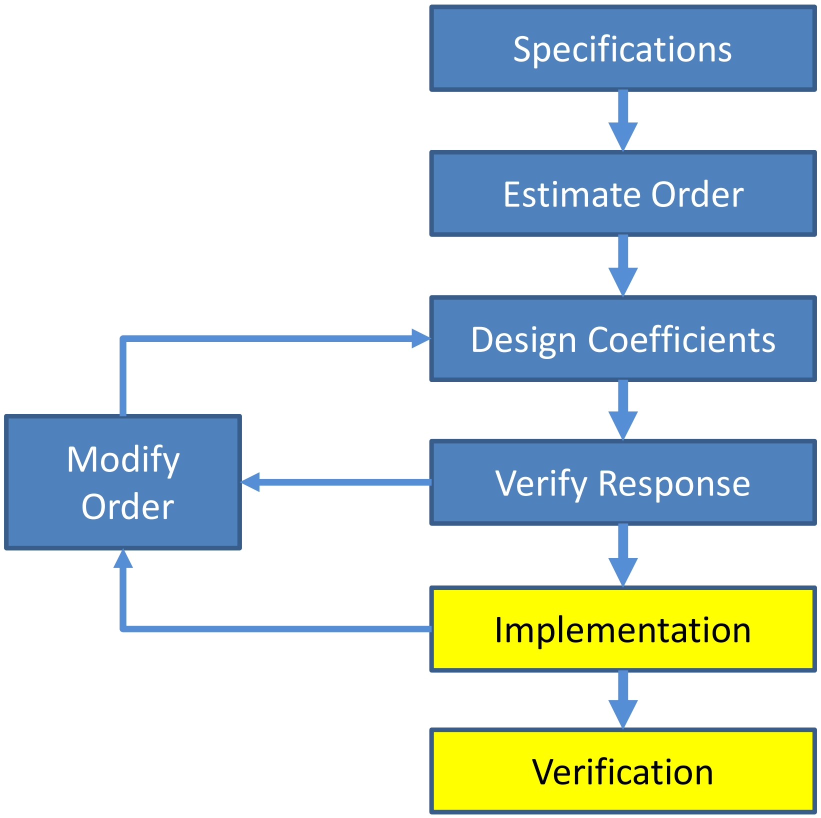






Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

FIR Design Flow



Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

FIR Implementation and Verification

Further details specific to FIR Implementation and Verification and much more are covered in the upcoming course:

"**DSP For Wireless Communications**"

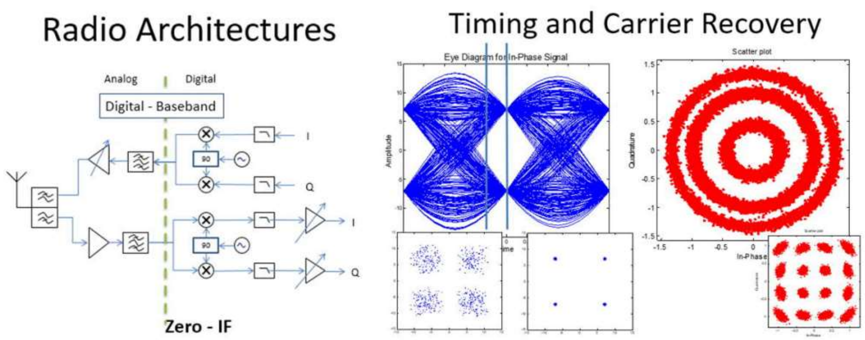
Starting October 10, 2024, through the Boston IEEE

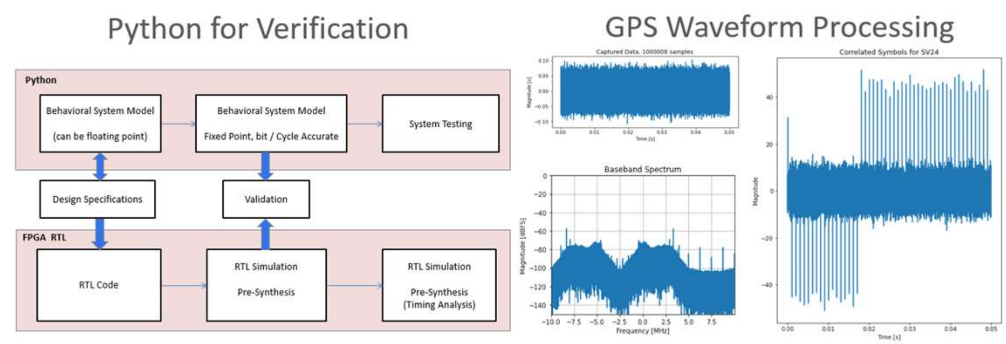
IEEE

Sign-up Link: https://ieeeboston.org/courses/

Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

Want more DSP?





"DSP For Software Radio"

"Python Applications for Digital Design and Signal Processing"

Sign-up Link: https://dsprelated.com/courses

Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

Python Tools

SciPy

matplotlib

NumPy

Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

References

Alan V. Oppenheim, Alan S. Willsky, with S. Hamid. *Signals and Systems*, Prentice Hall

fred harris, "On the use of windows for harmonic analysis with the discrete Fourier transform," in *Proceedings of the IEEE*, vol. 66, no. 1, pp. 51-83, Jan. 1978, doi: 10.1109/PROC.1978.10837.

Ronald W. Schafer, John R. Buck. *Discrete-Time Signal Processing*, Prentice-Hall Signal Processing Series by Alan V. Oppenheim

John G. Proakis, Dimitris K Manolakis. *Digital Signal Processing: Principles, Algorithms and Applications*, Prentice Hall

Paulo S. R. Diniz, Eduardo A.B. da Silva, Sergio L. Netto, *Digital Siganl Processing, System Analysis and Design*, Cambridge University Press

S.K. Mitra (1998). *Digital Signal Processing: A Computer-Based Approach,* McGraw-Hill, New York, NY

T.W. Parks and C.S. Burrus (1987). *Digital Filter Design,* New York: Wiley Interscience

Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

BACK-UP SLIDES

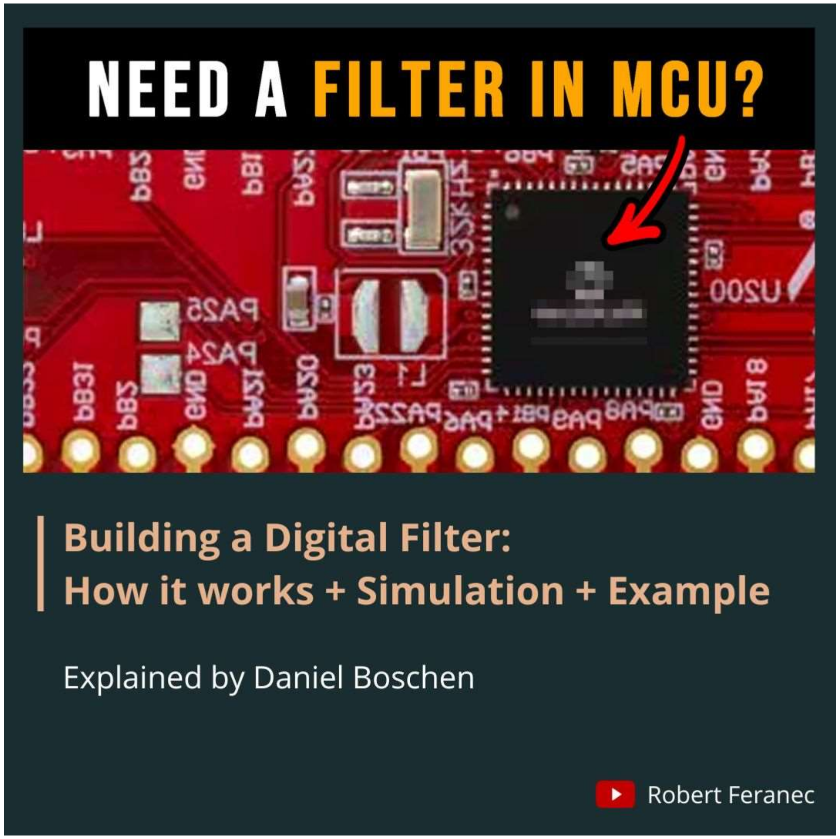
Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

IIR Notch Filter on a SAM-E51 Processor

A very simple explanation and demo of digital filters.

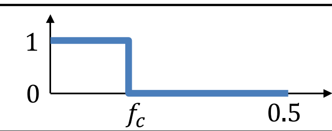
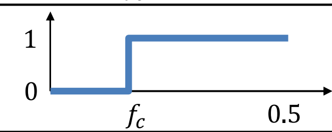
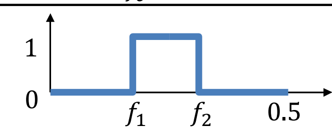
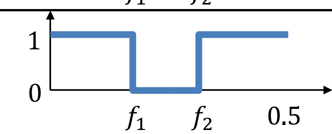
https://www.youtube.com/watch?v=Aq_SOvR1Sxs&t=1s

(or Google "Robert Feranec Boschen Filter")



Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

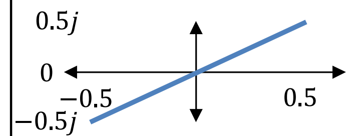
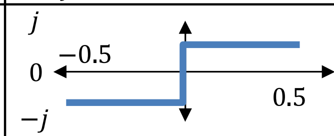
Impulse Responses for Ideal Filters

Filter Type	Impulse Response		Frequency Response $f_s = 1$
Lowpass, $h_{LP}[fc, n]$	$2f_c \mathrm{sinc}(2f_c n)$,	$-\infty \leq n \leq \infty$	
Highpass, $h_{HP}[fc, n]$	$h_{HP}[f_c, n] = h_{LP}[(0.5 - fc), n](-1)^n$	$-\infty \leq n \leq \infty$	
Bandpass, $h_{BP}[fc, n]$	$h_{LP}[f_2, n] - h_{LP}[f_1, n]$,	$-\infty \leq n \leq \infty$	
Bandstop, $h_{BS}[fc, n]$	$1 - 2(f_2 - f_1)$, $h_{LP}[f_1, n] - h_{LP}[f_2, n]$,	$n = 0$ $n \neq 0$	

$$\mathrm{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$$

Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

Impulse Responses for Ideal Filters

Filter Type	Impulse Response		Frequency Response $f_s = 1$
Differentiator, $h_d[n]$	$0,$ $\dfrac{(-1)^n}{n},$	$n = 0$ $n \neq 0$	
Hilbert, $h_H[n]$	$0,$ $\dfrac{1-(-1)^n}{\pi n},$	$n = 0$ $n \neq 0$	

Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

Estimate Beta and Order

Estimate of number of taps and beta using Kaiser Window:

$$\alpha_s = -20\log 10(\delta_2)$$

$$\begin{aligned}\beta &= 0.1102(\alpha_s - 8.7) \text{ for } \alpha_s > 50 \\ &= 0.5842(\alpha_s - 21)^{0.4} + 0.07886(\alpha_s - 21) \text{ for } 21 < \alpha_s \le 50 \\ &= 0 \text{ for } \alpha_s < 21\end{aligned}$$

$$N = \frac{\alpha_s - 8}{2.285\Delta\omega}$$

"Digital Signal Processing – A Computer Based Approach" – Sanjit K Mitra

Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

Using Parks-McClellan to Design Non-Linear Phase

From: https://dspguru.com/dsp/tricks/using-parks-mcclellan-to-design-non-linear-phase-fir-filter/

DSP Trick: Using Parks-McClellan to Design a Non-Linear Phase FIR Filter
From: ericj@primenet.com.nospam (Eric Jacobsen) Subject: DSP Trick - Using P-M to design a non-linear phase FIR filter. Date: 23 Oct 1999 00:00:00 GMT Newsgroups: comp.dsp THIS WORK IS PLACED IN THE PUBLIC DOMAIN
It is possible to use the Parks-McClellan algorithm to design FIR filters with non-linear phase response. For example, a FIR filter with the equivalent response of a Butterworth filter can be designed using the P-M routine.
First, take a common Butterworth description like that in Parks and Burrus where H(x) is a complex function. Create two PM input grids using the real and imaginary components of the Butterworth response. Use PM (or "remez" or whatever it's called on your favorite system) to design two FIR filters using the respective input grids, but turn the 'Hilbert' switch on for the one derived from the imaginary component. Sum the results, i.e., add together the nth coefficients of each filter to create a single N-tap filter.
The resulting FIR filter (assuming you've done your job to make sure everything converges) will have the desired response from the original expression used to generate the PM input grids.
Eric Jacobsen Minister of Algorithms

Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

Linear Phase FIR Types

Type	Length	Symmetry	Filter	Example
I	Odd	Symmetric	Any	[1 2 1]
II	Even	Symmetric	Not High	[1 2 2 1]
III	Odd	Anti-Symm	Band	[1 0 -1]
IV	Even	Anti-Symm	Not Low	[1 2 -2 -1]

Copyright © 2024, Dan Boschen, CC BY-NC-ND 4.0

Title Banner for Jupyter