FIR filter design with Parks-McClellan Remez

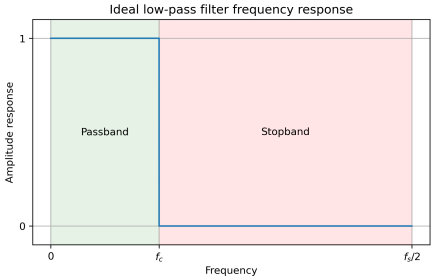
Daniel Estévez

1 February 2026
FOSDEM, Brussels

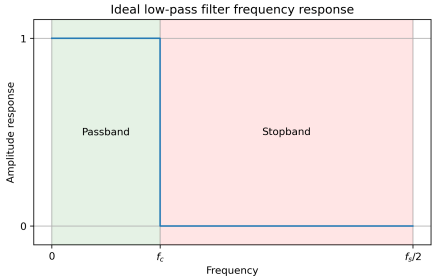
- Who knows the Parks-McClellan (Remez) filter design algorithm?

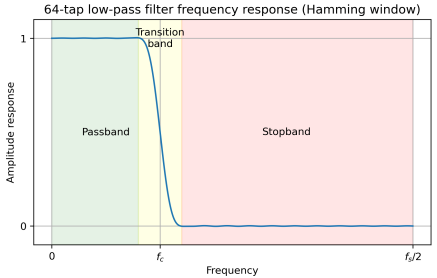
- Who knows the Parks-McClellan (Remez) filter design algorithm?
- Who does `pip install pm-remez`?

Motivation: designing a low pass filter

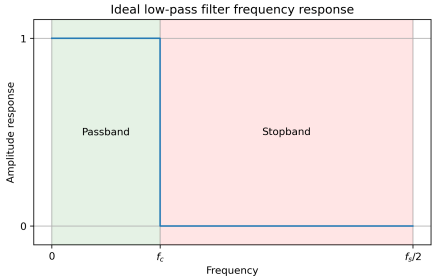


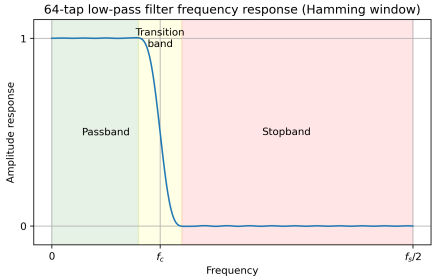
Motivation: designing a low pass filter

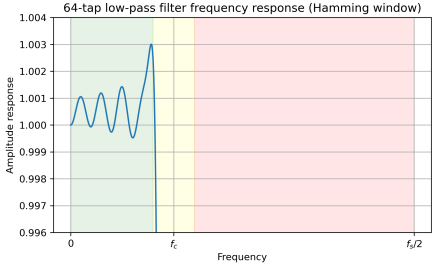


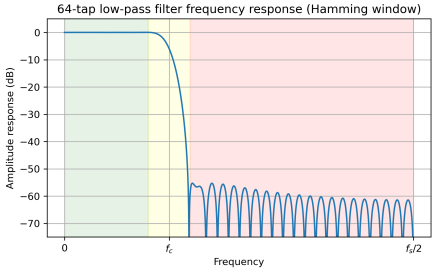


Motivation: designing a low pass filter







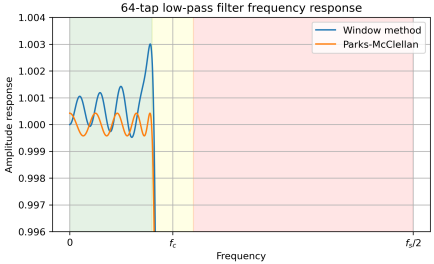


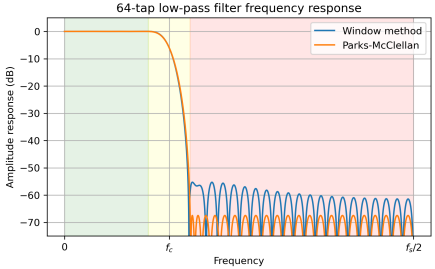
Optimal filter design

Optimization problem: find the FIR with N taps that minimizes

$$\max\{|H(f) - D(f)| : f \in \text{Passband} \cup \text{Stopband}\}.$$

Here $D(f)$ is the desired response: $D(f) = 1$ for $f \in \text{Passband}$, and $D(f) = 0$ for $f \in \text{Stopband}$.





Parks-McClellan algorithm

An algorithm that solves the following optimization problem. Find an FIR with real taps and either even or odd impulse response symmetry that minimizes

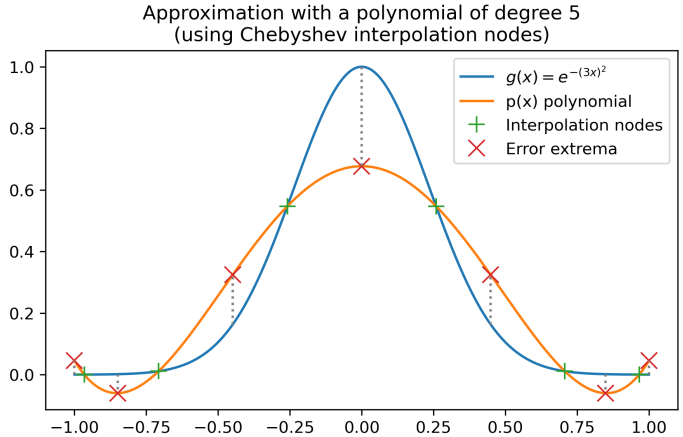
$$\|E\| = \max_{f \in F} E(f) = \max_{f \in F} W(f)|H(f) - D(f)|,$$

where $W(f) > 0$ is a weight function, $D(f) \in \mathbb{R}$ is the desired response, and F is a finite union of disjoint subintervals of $[0, f_s/2]$.

Note: the conditions on the FIR are equivalent to it having linear phase.

A detour: polynomial approximation

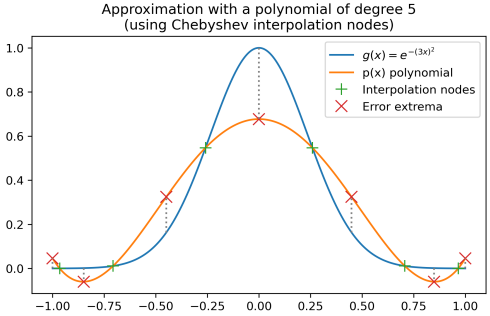
Chebyshev's equioscillation theorem: A polynomial p of degree $\leq n$ is the best uniform approximation to a continuous function g on an interval $[a, b]$ if and only if there are $n + 2$ points x_j in the interval such that $p(x_j) - g(x_j) = \pm\|p - g\|$ with alternating sign.

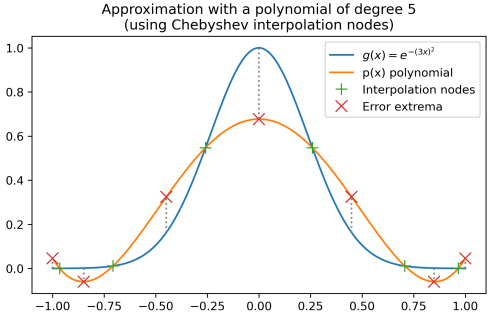


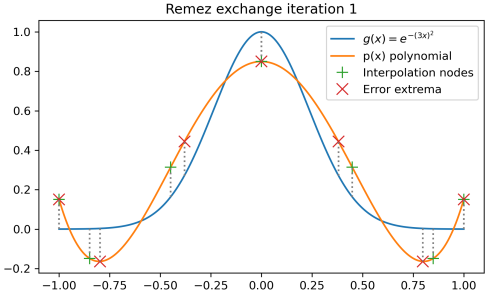
Remez exchange algorithm

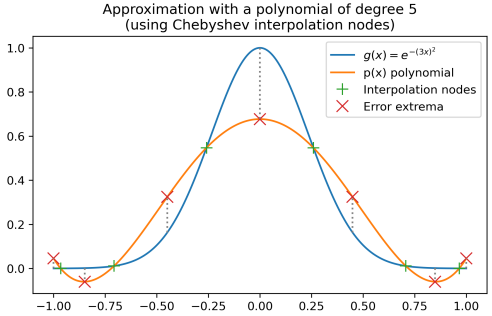
How do we compute this best uniform polynomial approximation?

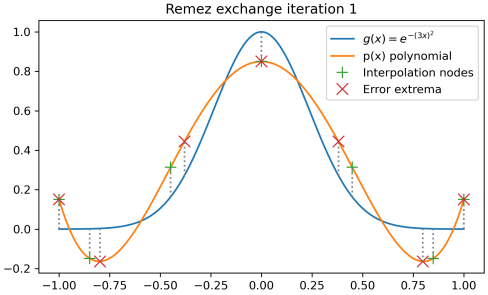
1. Start with $n+2$ points $x_j \in [a, b]$.
2. Solve the linear system $b_0 + b_1 x_j + b_2 x_j^2 + \cdots + b_n x_j^n + (-1)^j E = g(x_j)$ to get $b_0, b_1, \ldots, b_n$ and E.
3. Put $p(x) = b_0 + b_1 x + \cdots + b_n x^n$.
4. Find the local extrema of the error $p(x) - g(x)$, replace the $n+2$ points x_j by these extrema and go to step 2. Iterate until converged.

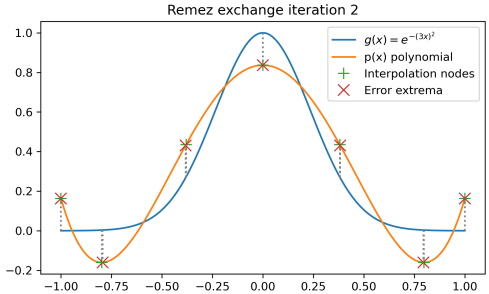

Approximation with a polynomial of degree 5
(using Chebyshev interpolation nodes)
$g(x) = e^{-(3x)^2}$
p(x) polynomial
Interpolation nodes
Error extrema
1.0
0.8
0.6
0.4
0.2
0.0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


Approximation with a polynomial of degree 5
(using Chebyshev interpolation nodes)
$g(x) = e^{-(3x)^2}$
p(x) polynomial
Interpolation nodes
Error extrema
1.0
0.8
0.6
0.4
0.2
0.0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


Remez exchange iteration 1
$g(x) = e^{-(3x)^2}$
p(x) polynomial
Interpolation nodes
Error extrema
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


Approximation with a polynomial of degree 5
(using Chebyshev interpolation nodes)
$g(x) = e^{-(3x)^2}$
p(x) polynomial
Interpolation nodes
Error extrema


Remez exchange iteration 1
$g(x) = e^{-(3x)^2}$
p(x) polynomial
Interpolation nodes
Error extrema


Remez exchange iteration 2
$g(x) = e^{-(3x)^2}$
p(x) polynomial
Interpolation nodes
Error extrema

What does polynomial approximation have to do with FIR filters?

Case of an odd length FIR with even symmetric impulse response (other cases are similar).

$$H(f) = \sum_{k=0}^{2n} a_k e^{-2\pi ikf} = e^{-2\pi inf} \sum_{k=-n}^{n} a_{k+n} e^{-2\pi ikf} = e^{-2\pi inf} \sum_{k=0}^{n} b_k \cos(2\pi kf).$$

Doing the change of variables $x = \cos(2\pi f)$, the function

$$\sum_{k=0}^{n} b_k \cos(2\pi kf)$$

can be written as a polynomial of degree n in x. (For instance, remember that $\cos(2\pi \cdot 2 \cdot f) = 2\cos^2(2\pi f) - 1 = 2x^2 - 1$).

Open source implementations of Parks-McClellan

- SciPy
  - C translation of original FORTRAN code $\Rightarrow$ horrible code
  - Piecewise constant response and weights
  - BSD license
  - Some convergence issues
- GNU Radio / Octave
  - C written from scratch in the 90s. Comments pointing out implementation problems (never fixed).
  - Piecewise constant response and weights
  - GPL-2.0-or-later license
  - Lots of convergence issues
- `https://github.com/sfilip/firpm` by S. Filip
  - Modern C++ using Eigen and GMP/MPFR
  - Piecewise constant response and weights
  - GPL-3.0-or-later until April 2024; BSD afterwards
  - Good convergence. Numerical methods described in a paper.

Open source implementations of Parks-McClellan

- `https://github.com/maia-sdr/pm-remez`
  - Rust using backends for linear algebra and higher float precision
  - Python package in PyPI (`pip install pm-remez`)
  - Arbitrary response and weights
  - Apache-2.0 or MIT license
  - Good convergence. Draws ideas from the S. Filip paper
  - Comes with documentation showing how to design different filter types

Potential convergence problems

In floating point arithmetic, the Remez exchange can fail to converge, specially for filters with a large number of taps

Some numerical optimizations:

- Barycentric Lagrange interpolation. Reduces numerical error in the interpolating polynomial.
- Chebyshev proxy root finding: Approximate error function by Chebyshev interpolating polynomial in a small interval. Find zeros of its derivative with an eigenvalue problem.
- Use higher precision (or arbitrary precision) floating point implementation

Practical filter design with pm-remez

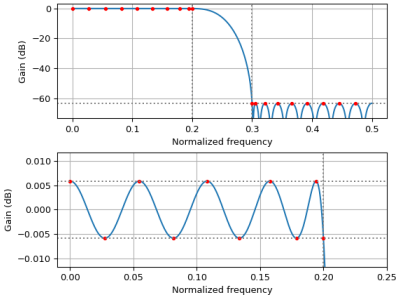
Get the code at `pm-remez.read-the-docs.io`

Basic anti-alias filter

Low-pass that avoids aliasign when decimating by D. Parameters:

- Decimation factor D (for instance $D = 2$)
- Transition bandwidth B (for instance $B = 0.2$)
- Number of taps N (for instance $N = 35$)

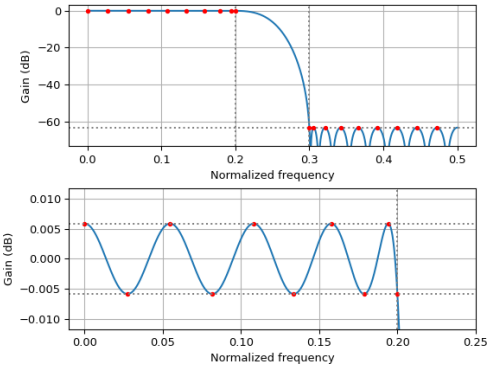
```
passband_end = 0.5 * (1 - B) / D
stopband_start = 0.5 * (1 + B) / D
pm_remez.remez(N, [0, passband_end, stopband_start, 0.5], [1, 0])
```



Improved anti-alias filter

Use a higher weight W in the stopband. For instance $W = 10$.
Rule of thumb: for 1% passband ripple, use $W = 10$ for -60 dB stopband attenuation, or $W = 100$ for -80 dB.

```
pm_remez.remez(N, [0, passband_end, stopband_start, 0.5], [1, 0],
               weight=[1, W])
```

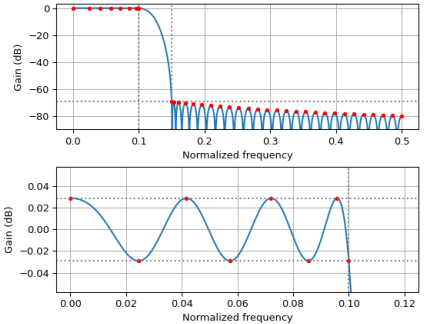


Improved anti-alias filter

Use a linearly increasing weight in the stopband to achieve a $1/f$ response. Useful for high decimation ratios.

```
sweight = (W, W * 0.5 / stopband_start)
pm_remez.remez(N, [0, passband_end, stopband_start, 0.5], [1, 0],
               weight=[1, sweight])
```

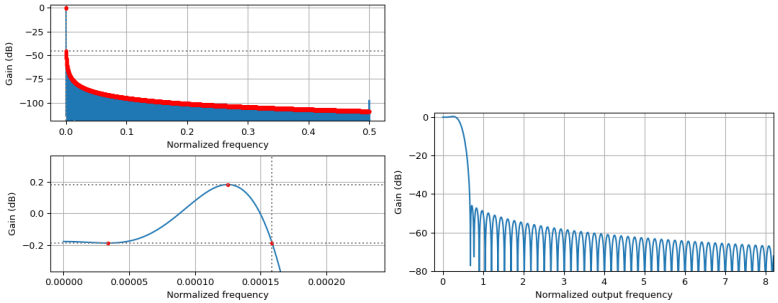
Example for $D = 4, B = 0.2, N = 67, W = 10$.



Polyphase filterbank prototype filter

A polyphase filterbank with C channels and T_c taps per channel (per arm) needs a decimator low-pass filter with $D = C$ and $N = CT_c$.

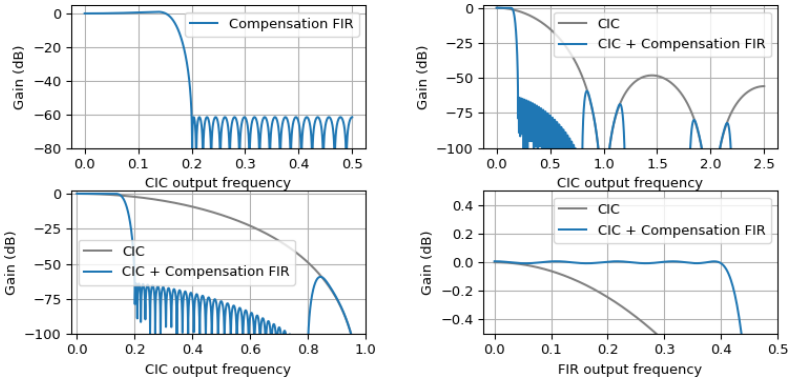
Example with $C = 2048$, $T_c = 6$, $B = 0.35$. The stopband weight W is tuned by hand to $W = 4$.



For larger C it is usually necessary to use a smaller C and then do linear interpolation of the taps.

CIC compensation filter

Example for 4-stage decimate-by-5 CIC and decimate-by-3 FIR with $B = 0.2, N = 53$.



M-stage CIC decimator by D_C frequency response:

$$\left|\frac{\sin(\pi f D_C)}{\sin(\pi f)}\right|^M = \left|\frac{\sin(\pi s)}{D_C \sin(\pi s/D_C)}\right|^M \approx \left|\frac{\sin(\pi s)}{\pi s}\right|^M \qquad (s = fD_C)$$

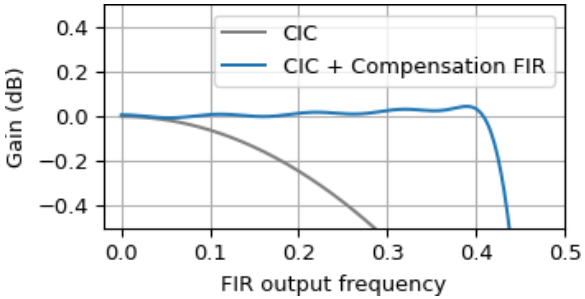
Compensation for a particular D_C:

```
desired = lambda f: (Dc*np.sin(np.pi*f/Dc)/np.sin(np.pi*f))**M
pm_remez.remez(N, [0, passband_end, stopband_start, 0.5],
                [desired, 0])
```

Compensation independent of D_C (decimation-independent):

```
desired = lambda f: (np.pi*f/np.sin(np.pi*f))**M
```

Decimation-independent decimate-by-3 FIR with $B = 0.2, N = 53$ used with 4-stage decimate-by-5 CIC.



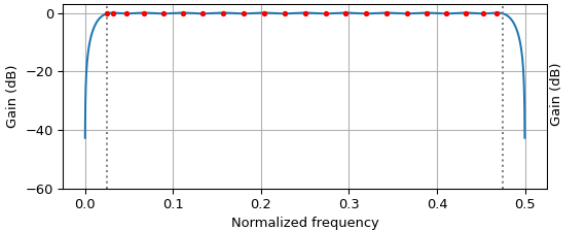
Hilbert filter

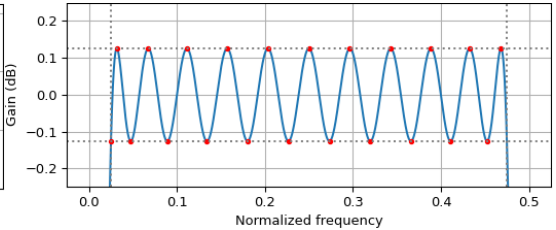
Ideal Hilbert filter: frequency response i for $f > 0$ and $-i$ for $f < 0$.

Practical FIR Hilbert filter:

- Odd taps and odd impulse response symmetry $\Rightarrow$ odd purely imaginary frequency response
- All-pass filter (but must have zero frequency response at DC and $f_s/2$)

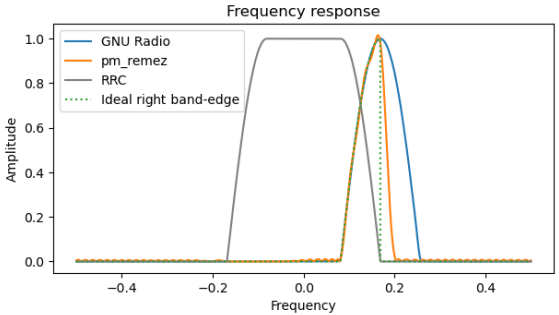
```
allpass_begin = 0.25 * B
allpass_end = 0.5 - 0.25 * B
pm_remez.remez(N, [allpass_begin, allpass_end], [1], symmetry='odd')
```

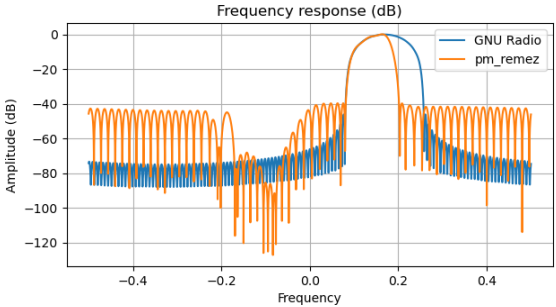




Other fun applications: band-edge filters

Comparison of fred harris / GNU Radio band-edge FIR and a pm-remez design (both 64 taps)

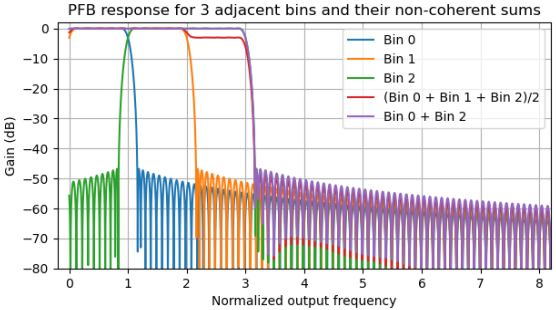


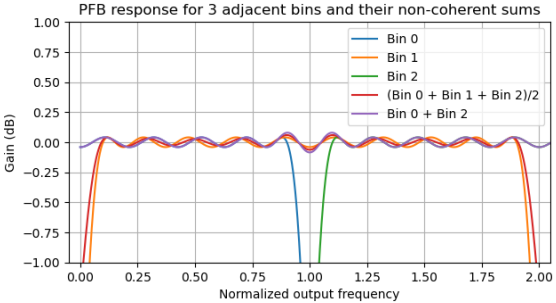


Remez design trick: design a filter with even response (sum of band-edges) and a filter with odd response (difference of band-edges).

Other fun applications: PFB for spectrum analysis

Problem statement: PFB for spectrum analysis with 50% overlap in the bins and response so that adjacent bins can be added (in power) to give a coarser resolution.





Remez design trick: tweak the cutoff frequency to achieve this property.