Half Band Filters

CSP / DSP LAB (Aug - Dec 2020)

Lecture - 4

Decimation & Interpolation

Decimation : LPF (anti-aliasing) + Downsampler

Interpolation : Upsampler + LPF (anti-imaging)

No of Computations in Convolution ??

- Multiplications — $(2N-1) * N$
- Additions — $(2N-1) * (N-1)$

Half Band Filter Properties

- Lowpass FIR filters with cut-off frequency of one-quarter of sampling frequency fs
- Even symmetry in time domain
- The passband and stopband bandwidths are equal, making these filters useful for decimation-by-2 and interpolation-by-2
- Almost half of the coefficients in time domain impulse response are zero

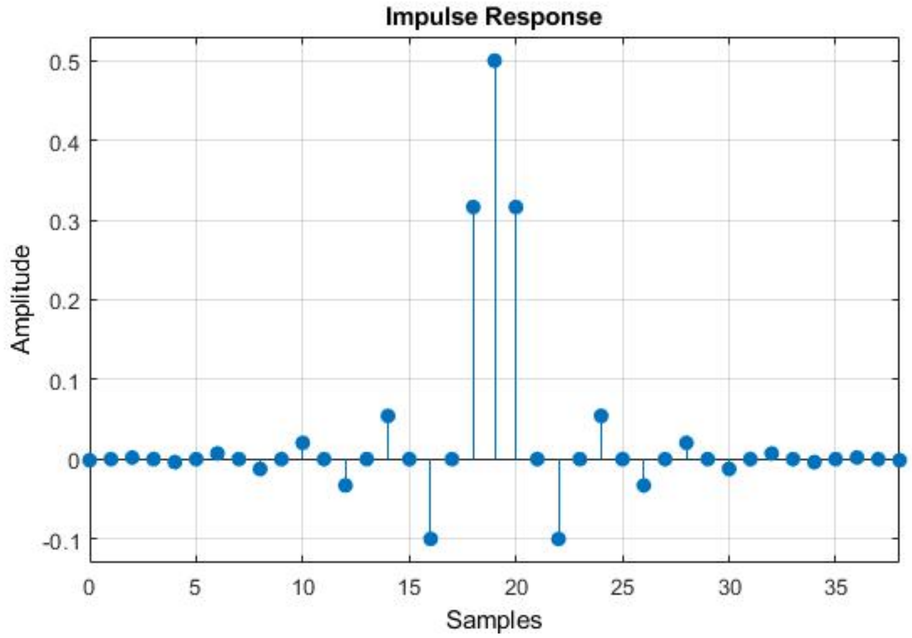
Zero Coefficients and Symmetry make them computationally efficient !

Ex : h[n] = [-0.0085, 0.0000, 0.2451, 0.5000, 0.2451, 0.0000, -0.0085]

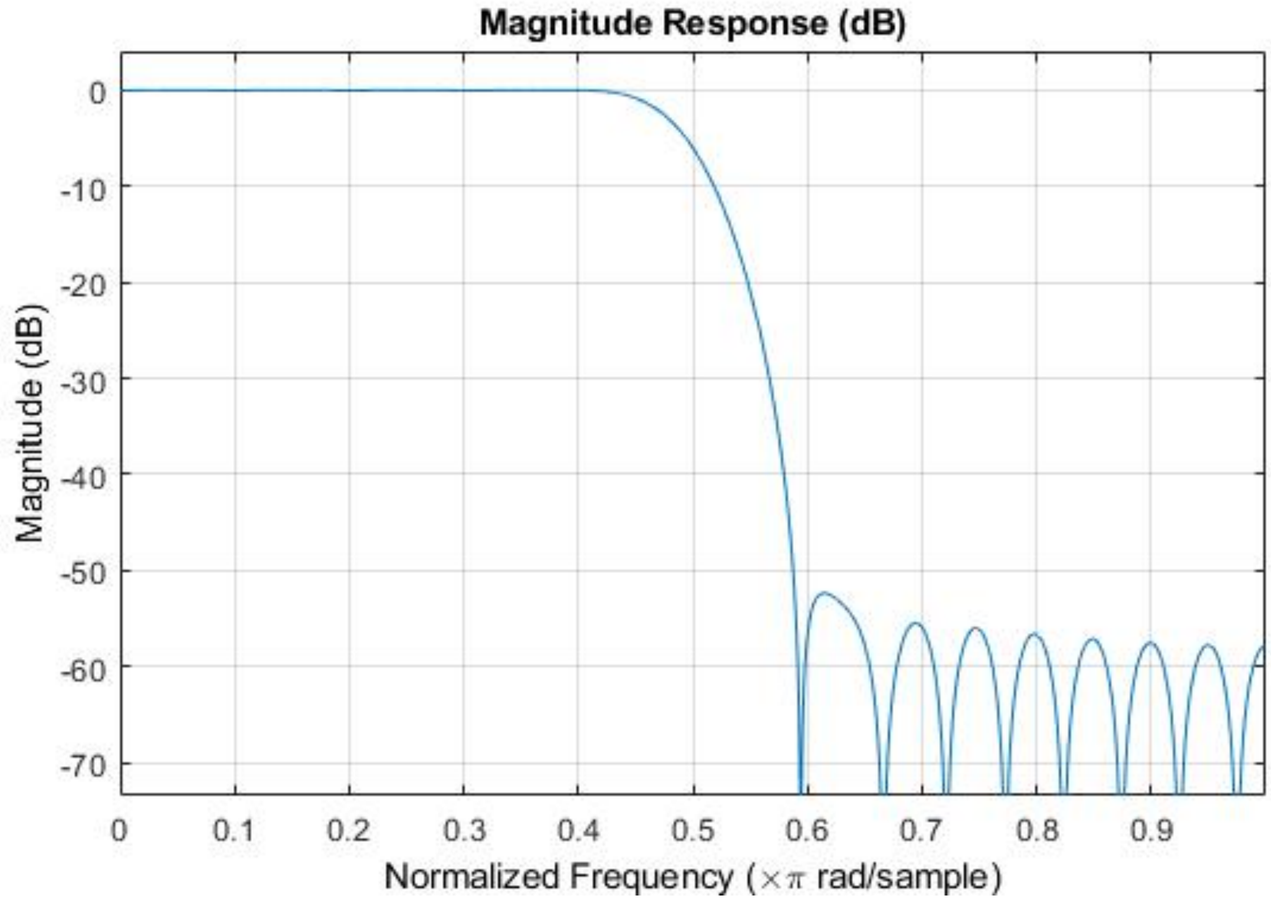
Half Band Filter

- HBF is a zero phase filter whose impulse response satisfies :

$$h(2n) = \begin{cases} c & n = 0 \\ 0 & n \neq 0 \end{cases}$$

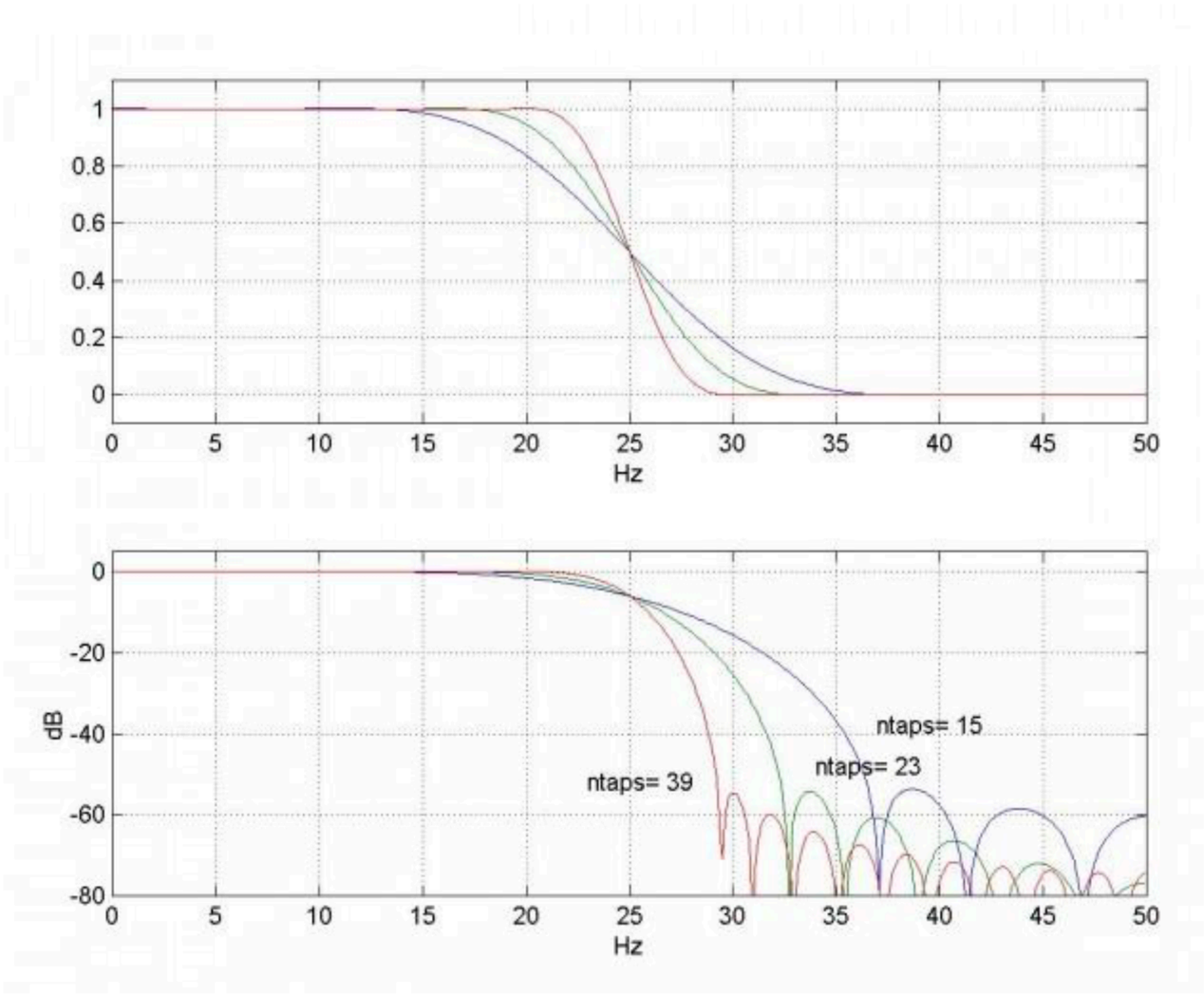


Magnitude Response of HBF



Magnitude Response of HBF

- top : Linear amplitude Scale
- bottom : dB amplitude Scale



Computational Efficiency in Decimation

- Compute only for middle lx samples of output of filter
- Symmetry
  - Calculate for only half of values ?
- Even values - Zero (Except for one value)
  - Ignore computation for values of filter where it is zero ?
- Downsampler :
  - Further ignore computing values which you anyways gonna remove at end!

Explanation

- Consider

x[n] = [0 1.0098, 0.1732, 1.5000, 0.6928, 0.4902, 0.0000, -0.4902, -0.6928, -1.5000, -0.1732, -1.0098, -0.0000, 1.0098, 0.1732, 1.5000, 0.6928, 0.4902, 0.0000, -0.4902, -0.6928, -1.5000, -0.1732, -1.0098, -0.0000, 1.0098, 0.1732, 1.5000, 0.6928, 0.4902, -0.0000, -0.4902, -0.6928, -1.5000, -0.1732, -1.0098]

h[n] = [-0.0085, 0.0000, 0.2451, 0.5000, 0.2451, 0.0000, -0.0085] : 7 Taps

Symmetry :

$y[6] = \sum x[6-m]*h[m]$

= x[6]*h[0] + x[5]*h[1] + x[4]*h[2] + x[3]*h[3] + x[2]*h[4] + x[1]*h[5] + x[0]*h[6]

= h[0]*(x[6] + x[0]) + h[1] *(x[5] + x[1]) + h[2]*(x[4] + x[2]) + x[3]*h[3]

Without HBF Properties -
No of multiplications : 7
No of Additions : 6

Even Values Zero :

y[6] = h[0]*(x[6] + x[0]) + h[1] *(x[5] + x[1]) + h[2]*(x[4] + x[2]) + x[3]*h[3]

= h[0]*(x[6] + x[0]) + h[2]*(x[4] + x[2]) + x[3]*h[3]

With HBF Properties -
No of multiplications : 3
No of Additions : 4

Downsampler (by 2) :

y[6] = h[0]*(x[6] + x[0]) + h[2]*(x[4] + x[2]) + x[3]*h[3] - remain same

But we need not calculate y[3] or y[5] !!

Computational Efficiency in Interpolation

- Upsampler :
  - Addition of zeros in input samples !!
- Symmetry
  - Calculate for only half of values !
- Even values - Zero (Except for one value)
  - Ignore computation for values of filter as well as input samples when it is zero !

Note : In case of interpolation ... Zero samples exist both in input as well as filter values

Thank You!