Fourier Transforms of Rectangular Functions and FreeDV Filtering

October 9, 2023

1 Introduction

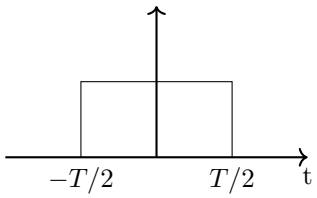
Using Fourier Transforms of Rectangular Functions, this document derives expressions for generating FreeDV band pass filter coefficients at run time, and efficient implementations of the filters.

2 Continuous Time Continuous Frequency

Consider a rectangular function in continuous time [1]:

$$rect(t) = \begin{cases} 1, & |t| <= T/2 \\ 0, & otherwise \end{cases} \tag{1}$$

Figure 1: Rectangular Function in Continuous Time



The continuous time, continuous frequency Fourier Transform is given by:

$$\begin{aligned} X(w) &= \int_{-\infty}^{\infty} rect(t)e^{-j\omega t}dt \\ &= \int_{-T/2}^{T/2} e^{-j\omega t}dt \\ &= \left[\frac{1}{-j\omega}e^{-j\omega t}\right]_{-T/2}^{T/2} \\ &= \frac{1}{-j\omega}\left[e^{-j\omega\frac{T}{2}} - e^{j\omega\frac{T}{2}}\right] \\ &= \frac{2sin(\omega T/2)}{\omega} \\ &= Tsinc\left(\frac{\omega T}{2}\right) \end{aligned} \tag{2}$$

where $sinc(x) = sin(x)/x$.

3 Discrete Time Continuous Frequency

Consider a rectangular function in discrete time:

$$rect(n) = \begin{cases} 1, & n = 0, 1, \ldots N-1 \\ 0, & otherwise \end{cases} \tag{3}$$

$$\begin{aligned} X(w) &= \sum_{n=-\infty}^{\infty} rect(n)e^{-j\omega n} \\ &= \sum_{n=0}^{N-1} e^{-j\omega n} \\ &= e^{-j\omega 0} + e^{-j\omega 1} + ... + e^{-j\omega(N-1)} \\ e^{-j\omega}X(w) &= e^{-j\omega} + e^{-j\omega 2} + e^{-j\omega 3} + ... + e^{-j\omega N} \\ X(w) - e^{-j\omega}X(w) &= 1 - e^{-j\omega N} \\ X(w) &= \frac{1 - e^{-j\omega N}}{1 - e^{-j\omega}} \\ \frac{e^{j\frac{\omega N}{2}}}{e^{j\frac{\omega}{2}}}X(w) &= \frac{e^{j\frac{\omega N}{2}}(1 - e^{-j\omega N})}{e^{j\frac{\omega}{2}}(1 - e^{-j\omega})} \\ e^{j\frac{\omega(N-1)}{2}}X(w) &= \frac{e^{j\frac{\omega N}{2}} - e^{-j\frac{\omega N}{2}}}{e^{j\frac{\omega}{2}} - e^{-j\frac{\omega}{2}}} \\ X(w) &= e^{-j\frac{\omega(N-1)}{2}}\frac{sin(\frac{\omega N}{2})}{sin(\frac{\omega}{2})} \end{aligned} \tag{4}$$

This can be interpreted as a complex term representing a linear phase shift (delay), multiplied by a real valued magnitude term. For small ω, the denominator term $sin(a) \approx a$, which results in a magnitude spectrum similar to the continuous time Fourier Transform (Equation 2).

Equation 4 can also be derived using the closed form expression for the sum of a geometric series:

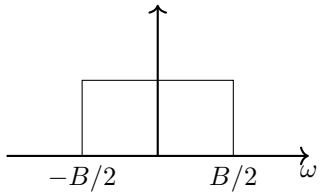
$$\sum_{k=0}^{n-1} ar^k = a\left(\frac{1-r^n}{1-r}\right) \tag{5}$$

4 Low Pass Filter Design

Consider a continuous frequency rectangular function:

$$H(\omega) = \begin{cases} 1, & |\omega| <= B/2 \\ 0, & otherwise \end{cases} \tag{6}$$

Figure 2: Rectangular Function in Continuous Frequency



$H(\omega)$ can be considered an ideal low pass filter for real input signals, with bandwidth $B/2$ radians. The discrete time, continuous frequency inverse Fourier

Transform is given by:

$$\begin{aligned} h(n) &= \frac{1}{2\pi}\int_{-\pi}^{+\pi} H(\omega)e^{j\omega n}d\omega \\ &= \frac{1}{2\pi}\int_{-B/2}^{+B/2} e^{j\omega n}d\omega \\ &= \frac{1}{2\pi jn}\left[e^{j\omega n}\right]_{-B/2}^{+B/2} \\ &= \frac{1}{2\pi jn}\left[e^{jnB/2} - e^{-jnB/2}\right] \\ &= \frac{sin\left(\frac{nB}{2}\right)}{\pi n} \\ &= \left(\frac{\frac{B}{2\pi}}{\frac{B}{2\pi}}\right)\frac{sin\left(\frac{nB}{2}\right)}{\pi n} \\ &= \frac{B}{2\pi}sinc\left(\frac{nB}{2}\right) \end{aligned} \tag{7}$$

Where $sinc(x) = sin(x)/x$. Note the normalised sinc function $sinc(x) = sin(\pi x)/(\pi x)$ is common in DSP software such as GNU Octave.

The infinite length even sequence $h(n)$ can be considered the time domain impulse response of an ideal low pass filter with cut off frequency $B/2$ radians. In practice filters are limited to N_{tap} taps by applying a tapered window. This results in non-ideal frequency response, rather than an ideal rectangular (brick wall) filter. A suitable length N_{tap} set of Hanning windowed filter coefficients can be found by:

$$\begin{aligned} c(i) &= h\left(i - \frac{N_{tap}-1}{2}\right)w(i) \quad i = 0 \ldots N_{tap}-1 \\ w(i) &= 0.5 - 0.5cos\left(\frac{2\pi}{N_{tap}-1}\right) \end{aligned} \tag{8}$$

This expression can be used when N_{tap} is odd or even.

One use case is filtering an OFDM modem signal which has significant side lobe energy at frequencies outside of the carriers. In this use case, we wish to have the filter -1dB point at a frequency just outside the carriers, in order to minimise distortion to the PSK constellation. For a given N_{tap}, the smallest $B/2$ can be found by experiment that produces acceptable distortion of the scatter diagram.

To filter an input signal $X(n)$ we convolve it with the impulse response $h(n)$:

$$y(n) = \sum_{k=0}^{N_{tap}-1} h(k)x(n-k) \tag{9}$$

5 Complex Band Pass Filter

Equation 6 can be considered a low pass filter of bandwidth $B/2$ radians for real valued signals, or a bandpass filter with centre frequency 0 radians and bandwidth B radians for analytic (single sided complex valued) signals. This suggests the low pass prototype filter can be shifted in frequency to construct a bandpass filter for analytic signals.

Given the low pass prototype coefficients $h(n)$, we wish to design a band pass filter:

$$H_\alpha(\omega) = \begin{cases} 1, & \alpha - B/2 <= \omega <= \alpha + B/2 \\ 0, & otherwise \end{cases} \tag{10}$$

Where α is the centre frequency of the bandpass filter of bandwidth B.

Figure 3: Bandpass Filter

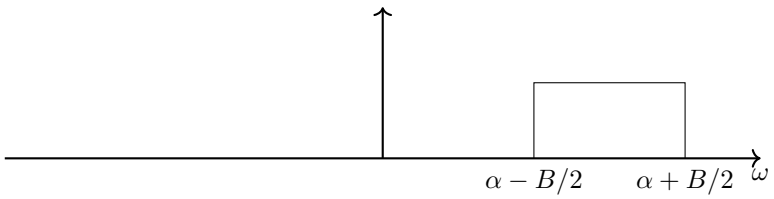
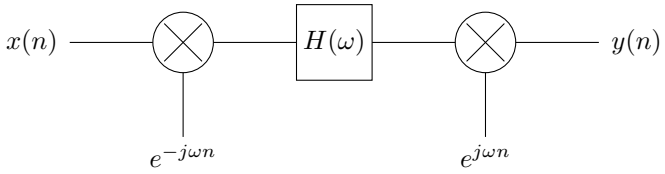


Figure 4: Bandpass Filter Signal Processing



Consider the time domain sequence of operations in Figure 4:

$$y(n) = e^{j\alpha n} \sum_{k=0}^{N_{tap}-1} h(k)x(n-k)e^{-j\alpha(n-k)} \tag{11}$$

Here we frequency shift the input energy in $x(n)$ centred on α down to 0, filter with the low pass filter $H(\omega)$, and shift the result back up to a centre frequency of α, effectively implementing the band pass filter in Equation 10. Re-arranging

Equation 11:

$$\begin{aligned} y(n) &= e^{j\alpha n} \sum_{k=0}^{N_{tap}-1} h(k)x(n-k)e^{-j\alpha n}e^{j\alpha k} \\ &= \sum_{k=0}^{N_{tap}-1} h(k)e^{j\alpha k}x(n-k) \\ &= \sum_{k=0}^{N_{tap}-1} h_\alpha(k)x(n-k) \\ h_\alpha(k) &= h(k)e^{j\alpha k} \end{aligned} \tag{12}$$

Note $h_\alpha(k)$ is complex, and can be pre-computed from $h(n)$ at initialisation time. An alternative approach to deriving Equation 12 would be to find $h_\alpha(k)$ directly from the inverse FT of (10).

5.1 CPU Load

Each operation in the summation of 12 is complex so requires $4N_{tap}$ Multiply Accumulates (MACs) to compute each $y(n)$, or for a block of N samples, $4N_{tap}N$ MACs.

In contrast, Equation 11 has real coefficients, so for complex $x(n)$ requires $2N_{tap}$ MACs to compute each $y(n)$, plus 4 MACs each for the up and down frequency shifts. We assume $e^{-j\alpha n}$ is stored as a look up table and requires no MACs. Alternatively it can be computed using a recursion with an additional 4 MACs/sample. The up and down conversions are conjugates so only one needs to be computed, the other can then be obtained by a sign change. For a block of N samples, this is a total of $(2N_{tap}+8)N$ MACs.

Table 11 presents the MACs required to process a 100ms ($N = 800$ samples at $Fs = 8000Hz$) block of samples. In this example the real coefficient algorithm in Equation 11 is the most efficient.

Algorithm	MACs
Real $h(k)$	166,400
Complex $h_\alpha(k)$	320,000

Table 1: Multiply Accumulates (MACs) to bandpass filter a $N = 800$ block of samples using a $N_{tap} = 100$ filter

6 Filter Implementation

Consider filtering a block of N samples:

$$y(n) = \sum_{k=0}^{N_{tap}-1} h(k)x(n-k) \quad n = 0, 1, \ldots N-1 \tag{13}$$

This requires a filter memory of previous $N_{tap} - 1$ samples of $x(n)$ to be maintained. For efficient implementation we wish to avoid managing this memory during the inner MAC, for example manipulating pointers or shifting samples in a delay line. An efficient algorithm is:

1. At initialisation time, set up an array of $N_{tap} - 1 + N$ samples, initialise the first $N_{tap} - 1$ samples to zero.
2. Copy the latest N input samples into the last N samples of the array
3. Evaluate Equation 13 over the last N samples of the array.
4. Update the filter memory by copying the last $N_{tap} - 1$ samples of the array to the start of the array

Steps 2-4 are repeated for each block of N samples.

7 Further Work

1. Octave script to compare and plot fir1() to 7. Show effect of windowing on frequency response.
2. Maths for sample rate conversion, zero insertion, decimation.
3. Rather than selecting $B/2$ by trial and error, we could find an expression for the windowed FT $X_w(\omega)$, and solve for $B/2$ given $|X_w(\omega)| = 10^{-1/20}$, where ω is the frequency of the desired -1dB point (e.g. just outside of the outermost OFDM carrier). This could be evaluated for any N_t, trading off filter performance (and hence the OFDM signal spurious mask) for CPU load.

References

[1] Rectangular Function. `https://en.wikipedia.org/wiki/Rectangular_function`.