# FIR Windowed Sinc Filter (Rectangular Window)

## Ideal Frequency Response of the Filter

1. The X-Axis of the filter is designed over the range -fs/2 to fs/2 over N points
2. N is chosen to be 201 - so that you have a 0 index in the IFFT -> 0-99(100 pts); and 101-200(100 pts) with a 0 at 0th index.
3. Fc is set at 50hz cut-off

*IFFT needs the input to be in the following format:*

*The input should be ordered in the same way as is returned by FFT, i.e.,*

* *a[0] should contain the zero frequency term,*
* *a[1:n//2] should contain the positive-frequency terms,*
* *a[n//2 + 1:] should contain the negative-frequency terms, in increasing order starting from the most negative frequency.*

1. ‘H’ is set to 1 for frequencies running from 0 to 50hz(indices) - over (N-1)/2 points.

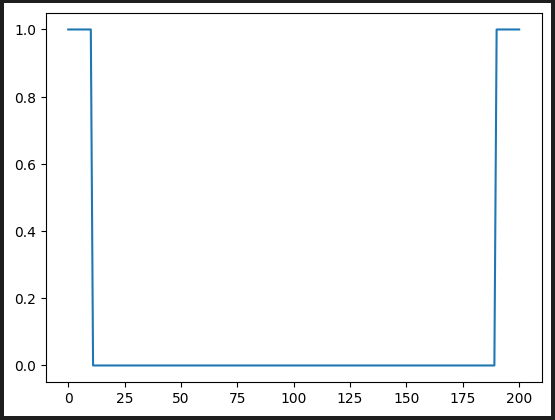
Eg: if N = 201;

N-1 = 200 ; (N-1)/2 = 1

freq = np.linspace(0,fs/2,(((N-1)//2) + 1))

Creates a frequency vector from 0 to fs/2 over 101 points where the 0th point is DC and the remaining 100 points are frequencies.

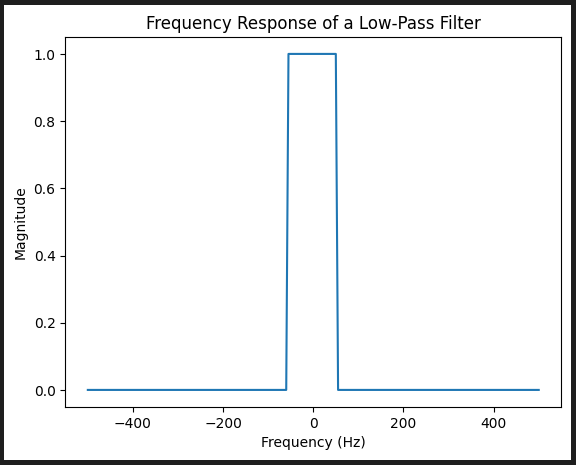
1. H = np.where(freq<=fc, 1, 0) : Sets the frequency 1 till fc and 0s the rest
2. Mirror H and concatenate - and [1:] ensures the DC(0th frequency) is not included again : H\_full = np.concatenate((H, H[::-1][1:]))

Hence H\_full would be 201 in samples, 0 at 0th frequency; 1 to 100 would be the positive frequencies and 101 to 200 would be positive frequencies mirrored without the DC.

Here the X-Axis is number of samples and Y-Axis is the magnitude.

plt.plot(freqs\_x,np.fft.fftshift(H\_full)) :

FFT shift and plot over frequencies - positive and negative.



## Plotting Impulse Response of an FIR Filter

1. Set up the time vector from -(N/2)/fs to (N/2)/fs

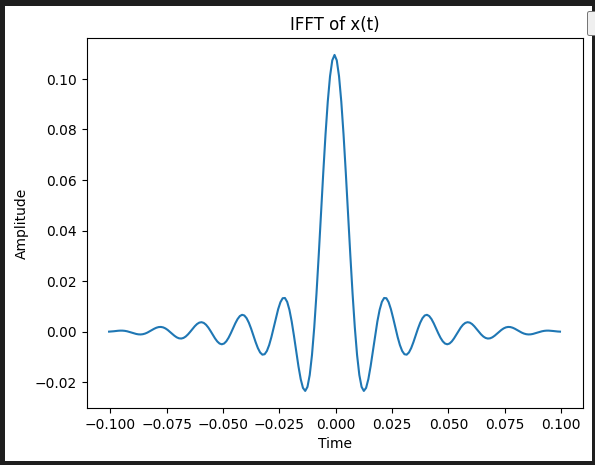
t = np.arange(-N/2,N/2)/fs

1. Take the IFFT of the Ideal frequency response and FFT shift to align 0 with 0 of the time-vector

xt = np.fft.fftshift(np.fft.ifft(H\_full))

print(xt[100])

0 centering is ensured by printing the amplitude of xt[(N-1)/2]



## Truncating the Impulse Response

1. Filter Order(M) = 50 will include M+1 points including the 0th hz.

M = 50 #Filter Order

1. Calculate the mid-point.

mid = (N-1)/2

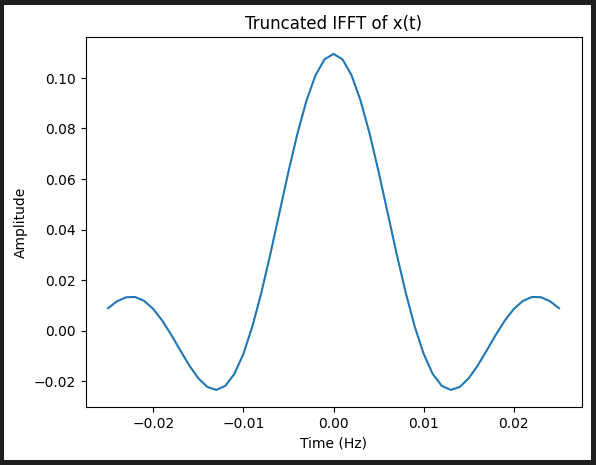
1. Truncate the impulse response to M+1 points, with M/2 points on each side of the 0 centre. (If M=50, length of the truncated Sinc will be 51)

truncated\_xt = xt[int(mid-(M/2)):int(mid+(M/2)+1)]

print(len(truncated\_xt))

1. Define the time vector from (-M/2 to M/2)/fs

t\_truncated = np.arange(-M/2,M/2+1)/fs



Even though - the sinc function is zero-centered, it’s impossible in Real-time to convolve this signal with our input signal. The sinc function has to be shifted to span from 0 to M. This isn’t explicitly done, yet the goal is achieved by convolving the signal with present and past values only.

plt.figure()

plt.plot(np.real(truncated\_xt))

plt.title('Truncated IFFT of x(t)')

plt.xlabel("Time (Hz)")

plt.ylabel("Amplitude")

plt.show()

## Extend the truncated signal to N samples by padding zeroes

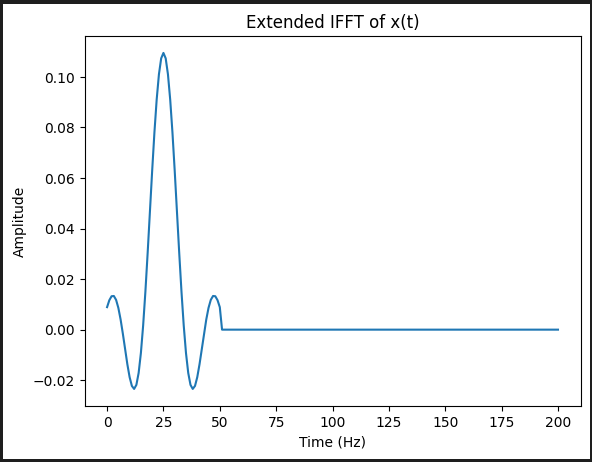
#extend the truncated xt to 201 samples

extended\_xt = np.zeros(N-(M+1)) #150

print(len(extended\_xt))

extended\_xt = np.concatenate((truncated\_xt, extended\_xt))

print(len(extended\_xt)) #201



## Frequency Response of the Truncated Impulse Response

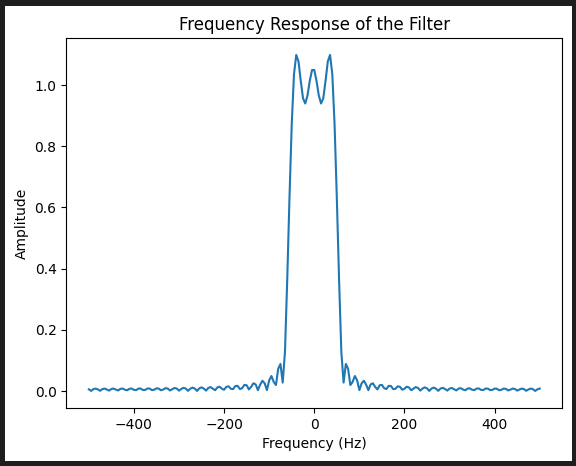
1. Take FFT of the truncated signal

f\_truncated = np.fft.fft(extended\_xt)

1. Set up the frequency vector from -fs/2 to fs/2 because the plotted response is FFT shifted.

freq = np.linspace(-fs/2,fs/2,N)

1. FFT shift the frequency response

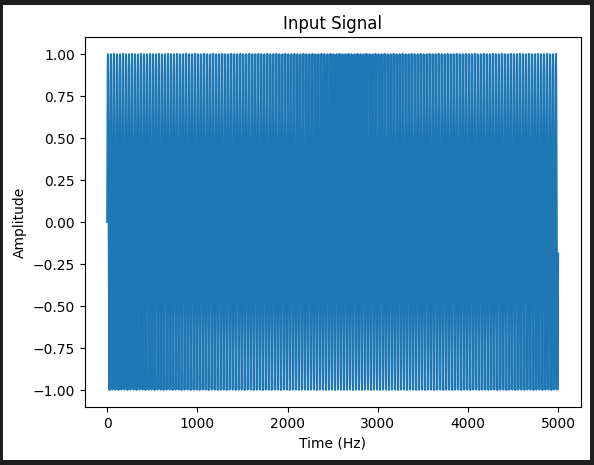


## Setup a 5s sine wave of a certain frequency - Frequencies within the pass band shouldn’t get attenuated.

t = np.arange(0, 5, 1/fs) # Time vector for 1 second

print(len(t))

signal = np.sin(2\*[np.pi](http://np.pi)\*30\*t)



## Filtered aka Convolved signal

y = np.convolve(signal, extended\_xt, mode='same')

