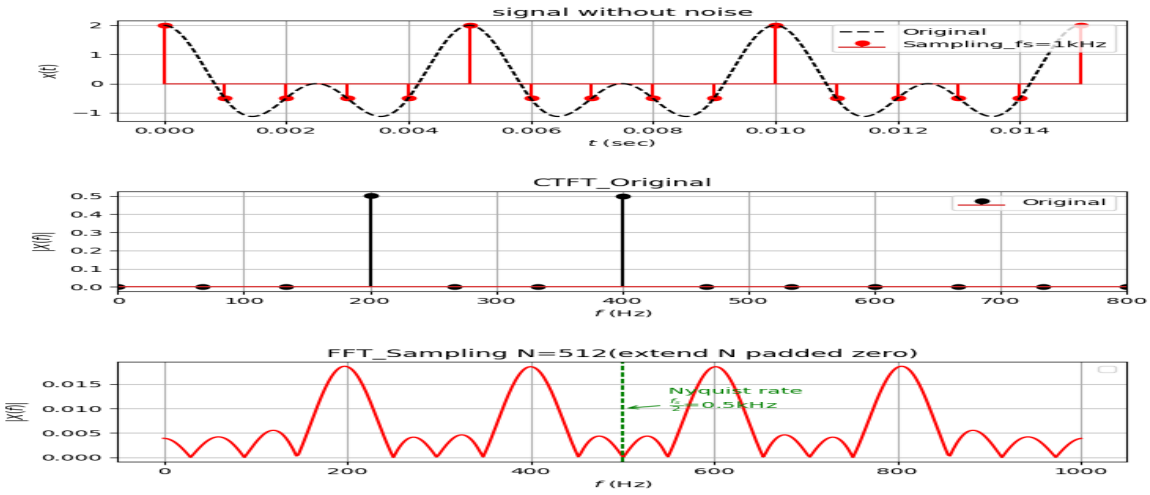
DSP 프로젝트

학과 : 전자공학과
이름 : 박정진
학번 : 2015104027

문제 해결

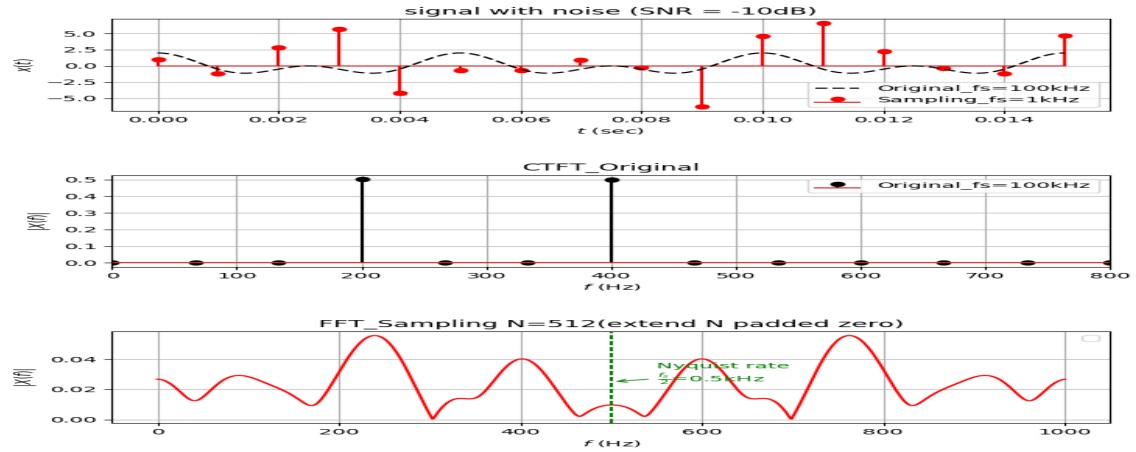
1. 잡음이 전혀 없는 경우 수신신호의 신호파형과 스펙트럼 특성

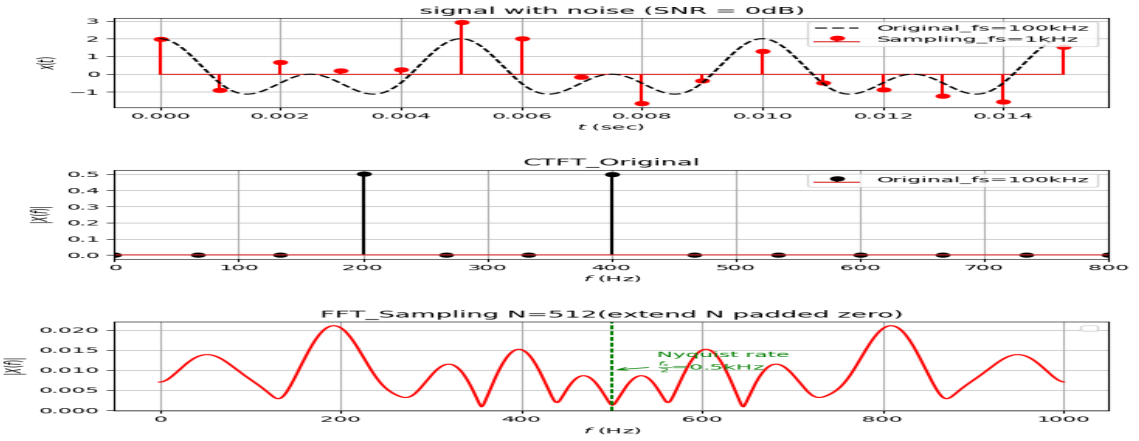
잡음이 전혀 없는 경우 두 개의 잠수함으로부터 반사된 신호는 $x(t) = \cos(2\pi f_1 t) + \cos(2\pi f_2 t)$ 일 것이다. 이 때 $f_1 = 200Hz, f_2 = 400Hz$ 라고 했을 때 이 신호의 주파수는 두 주파수의 최대공약수인 200Hz 가 될 것이고 따라서 주기는 5ms 이다. 그래서 0~15ms 로 3 주기의 입력을 받았고, sampling frequency $f_s = 1kHz$에 의해 샘플링 개수는 16 개를 받을 것이다.
실제 신호를 그리려 코드 상으로 $f_s = 100kHz$로 원 신호에 거의 가깝게 그렸고 그것에 따라 python library 중 하나인 numpy.fft 에 있는 fft 함수를 이용해 스펙트럼을 그렸다. FFT 는 DFT 의 연산을 더욱 빠르게 하는 계산으로 그 결과를 분석해보면 $X(f) = T_s X[k] = \frac{1}{N} X[k],\ f_a = \frac{k}{N} f_s$ 를 따르는 것을 알 수 있다. 원래 이론적으로는 FFT 결과에 $\frac{2}{N}$ 을 해야하는데 그 반 값을 한 이유는 컴퓨터 상에서는 주파수의 양수 부분만 계산하므로 주파수의 음수 부분이 계산이 안되어 있으므로 반쪽의 결과만 나온다. 만약 주파수의 음수 부분도 계산하고 싶다면 numpy.fft 라이브러리에 있는 fftshift 라는 함수를 사용하면 fft 의 결과가 주파수의 음수 부분 즉 $[-\pi, \pi]$ 의 결과 값이 출력된다.
다시 본론으로 돌아와 샘플링한 데이터를 FFT 했을 때 샘플링 개수가 16 개인 관계로 너무 적다 판단하여 FFT 의 샘플링에 zero 값을 넣어주어 샘플링 개수가 512 개가 되도록 zero padding 을 하였다. (이후 filter 에 넣을 때는 샘플링 개수 16 개를 넣었다.) FFT 의 frequency domain spectrum 분석결과 200, 400Hz 에서의 값이 가장 크게 나왔음을 알 수 있고 시뮬레이션 결과 FFT 를 할 때 Zero padding 을 많이 하면 할 수록(혹은 Sampling frequency 가 높아 샘플링 개수가 많으면) FFT 스펙트럼은 원 신호의 스펙트럼의 모양, 값과 더 비슷해 지는 것을 확인 할 수 있었다. 또한 FFT 의 스펙트럼을 보면 정확히 Nyquist rate 인 $\frac{f_s}{2}$를 기준으로 mirroring 결과가 나오는 것을 볼 수 있다.

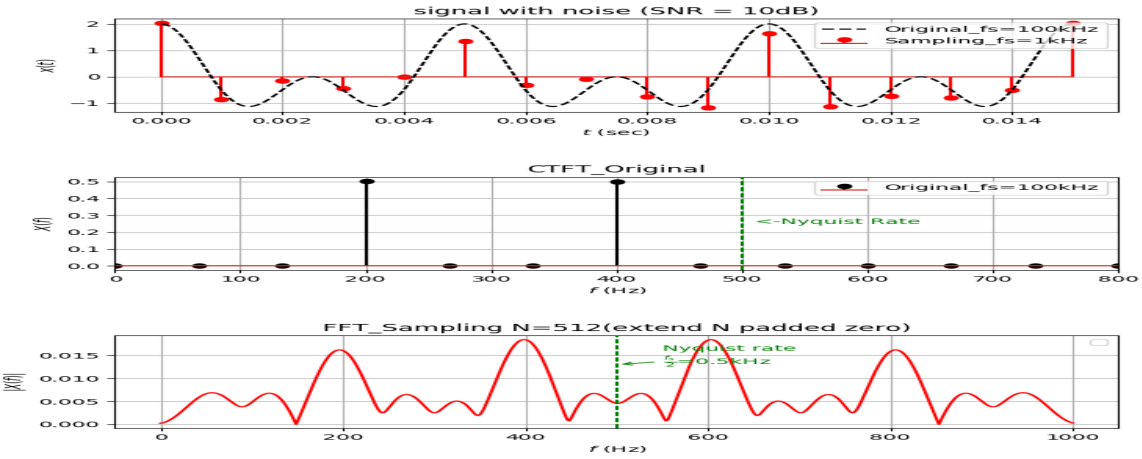


2. 신호 대 잡음비가 -10dB, 0dB 및 10dB 인 경우의 수신신호의 스펙트럼 특성 차이

1 의 문제와는 달리 수신신호에 Gaussian noise 를 더하는 문제였는데 numpy.random 라이브러리에 normal 함수를 이용하여 gaussian noise 를 생성했다. 당연하게도 SNR 이 커질 수록 신호는 원 신호 $x(t) = \cos(2\pi f_1 t) + \cos(2\pi f_2 t)$ 와 비슷해 질 것이고 스펙트럼 또한 마찬 가지 일 것이다.

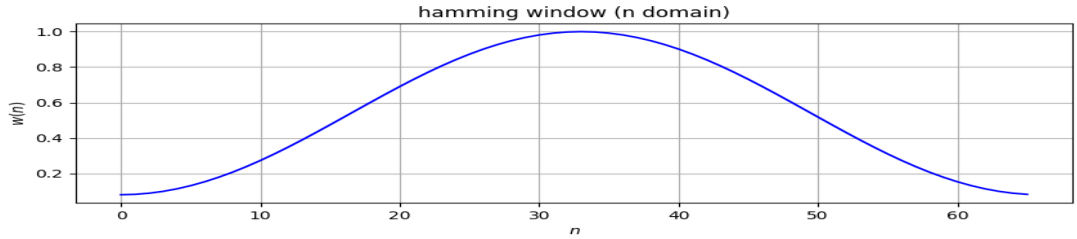


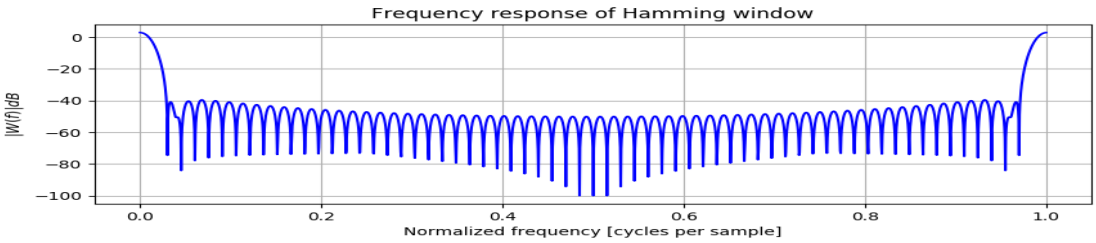




3. 두 목표물(신호)를 분리하기 위한 FIR 저역통과필터(LPF) 및 대역통과필터(BPF)를 설계

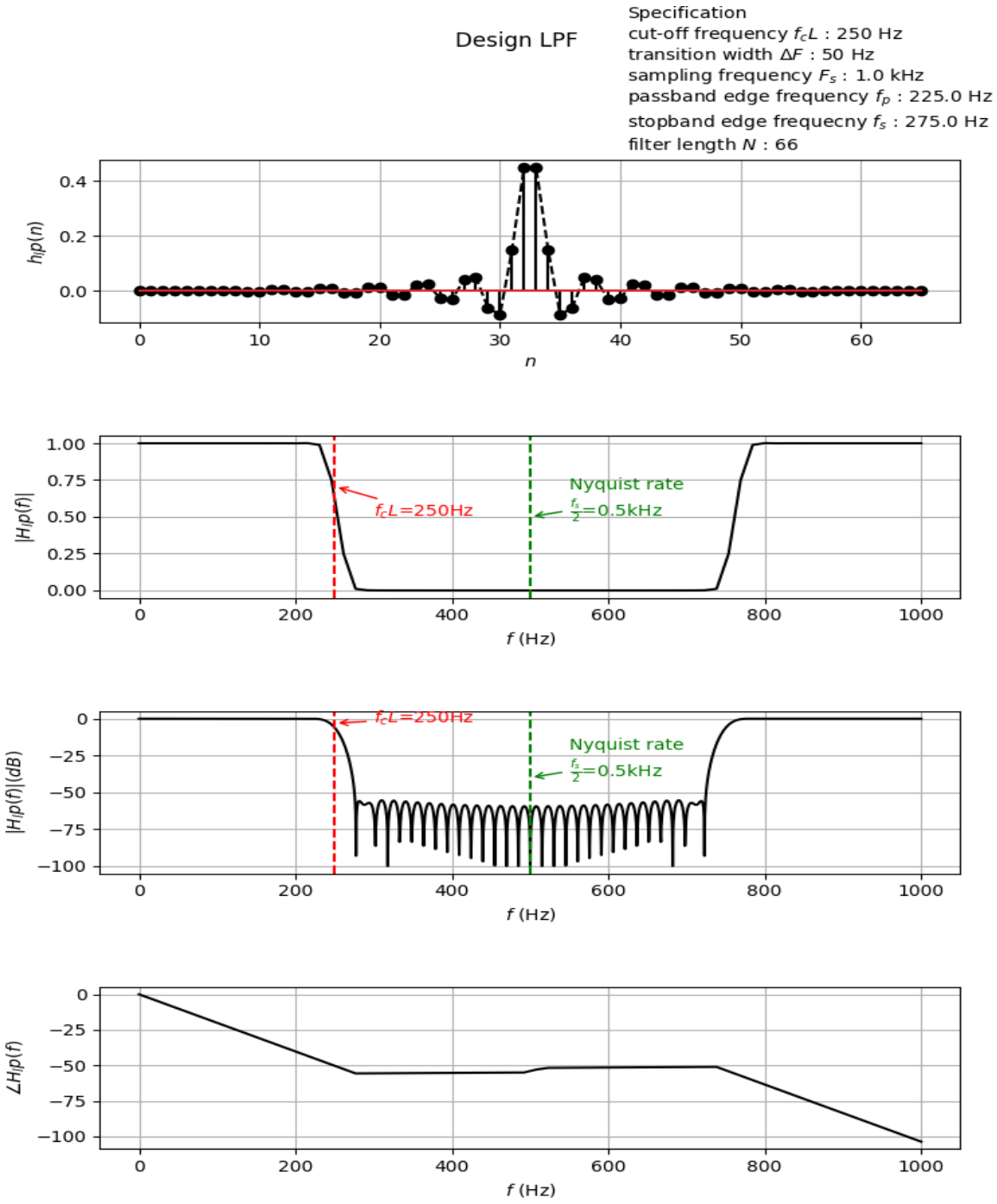
LPF 로 200Hz 의 cos 을, BPF 로 400Hz cos 을 분리하려고 한다. Filter 는 window method 를 사용했고 window type 중 가장 대중적이고 많이 쓰이는 hamming window 를 사용했다. hamming window FIR 필터는 python library scipy.signal.firwin 함수를 이용하여 손쉽게 구현할 수 있었고 이 함수는 window 의 string 만 입력하면 그 string 에 맞는 window 를 이용해 FIR filter 를 만들어준다. 만약 window 만 갖고 오고 싶다면 scipy.signal.get_window 함수를 사용하면 되고 위와 마찬가지로 window 의 이름만 입력하면 window 를 만들어준다.





이후 LPF 를 재사용하기 위하여(BPF 를 설계할 때 LPF 클래스 생성) 클래스를 이용하여 window type, cutoff frequency, sampling frequency, transition width 를 입력으로 받아 coefficient 를 바꿀 때 유저의 입력 값만 바꾸면 바로 사용할 수 있도록 코드를 구성했다. filter 의 specification 은 교수님이 주신 ppt 파일을 참고하여 설계하였다.

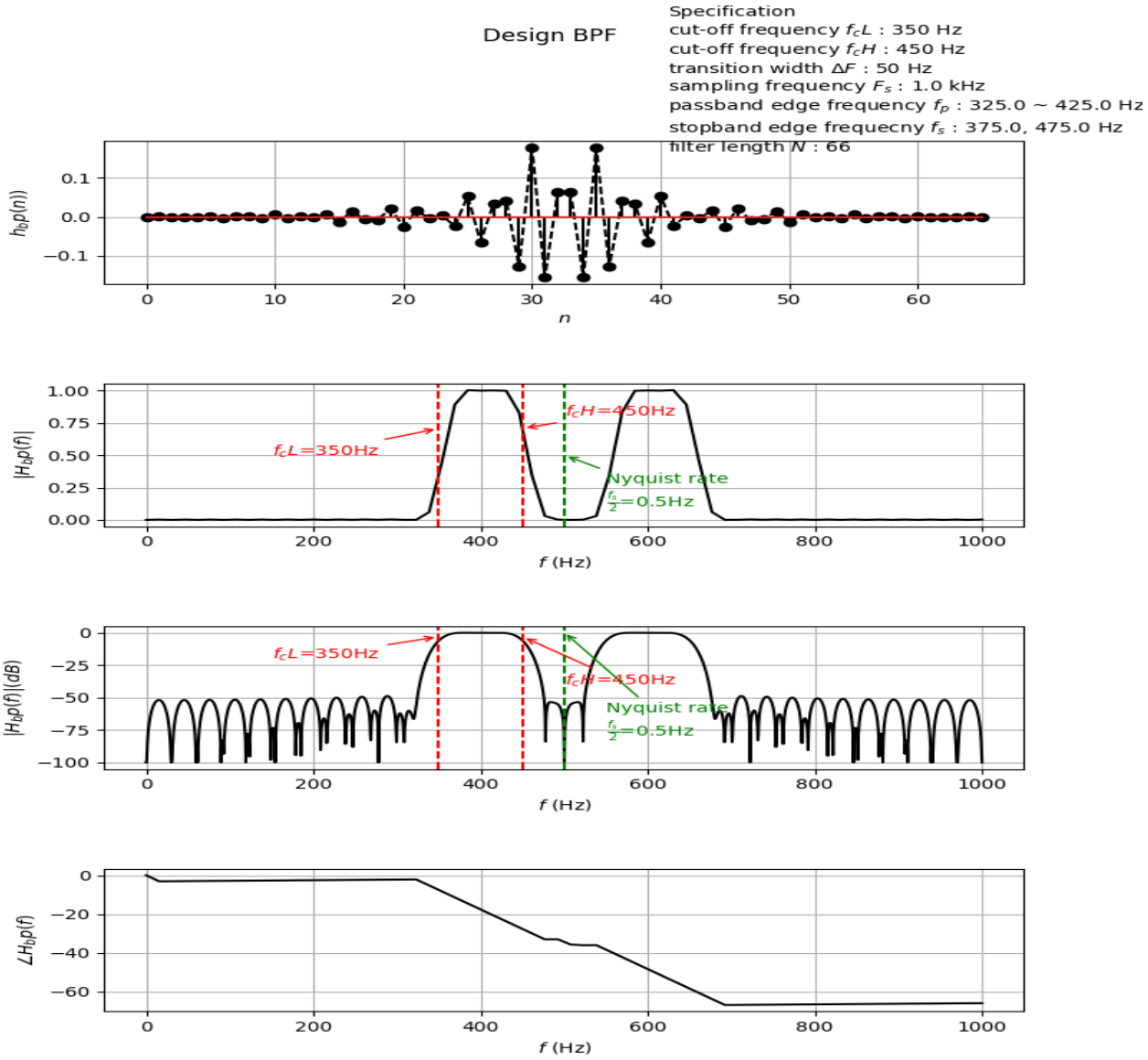
1. LPF 설계 : cut-off frequency 250Hz, transition width 50Hz



LPF 의 코드를 보면 위의 입력 값을 넣어주면 ppt 에 나와있는 table 을 기준으로 모든 기준이 맞춰지는데 그 값들은 그림 상단 우측에 자동으로 기록된다.
코드를 만들 때 filter length N 의 값을 even 으로 만들었는데 positive symmetry 를 형성하여 $\theta(w) = -\alpha w T_s$ 인 linear phase response 가 나오는 것을 확인 했다. 이 특성은 이후 신호를 필터에 통과시키고 난 후 출력 신호의 형태에서 확실히 확인 할 수 있다.

2. BPF 설계 :
cut-off low frequency 350Hz, cut-off high frequency 450Hz,
transition width 50Hz

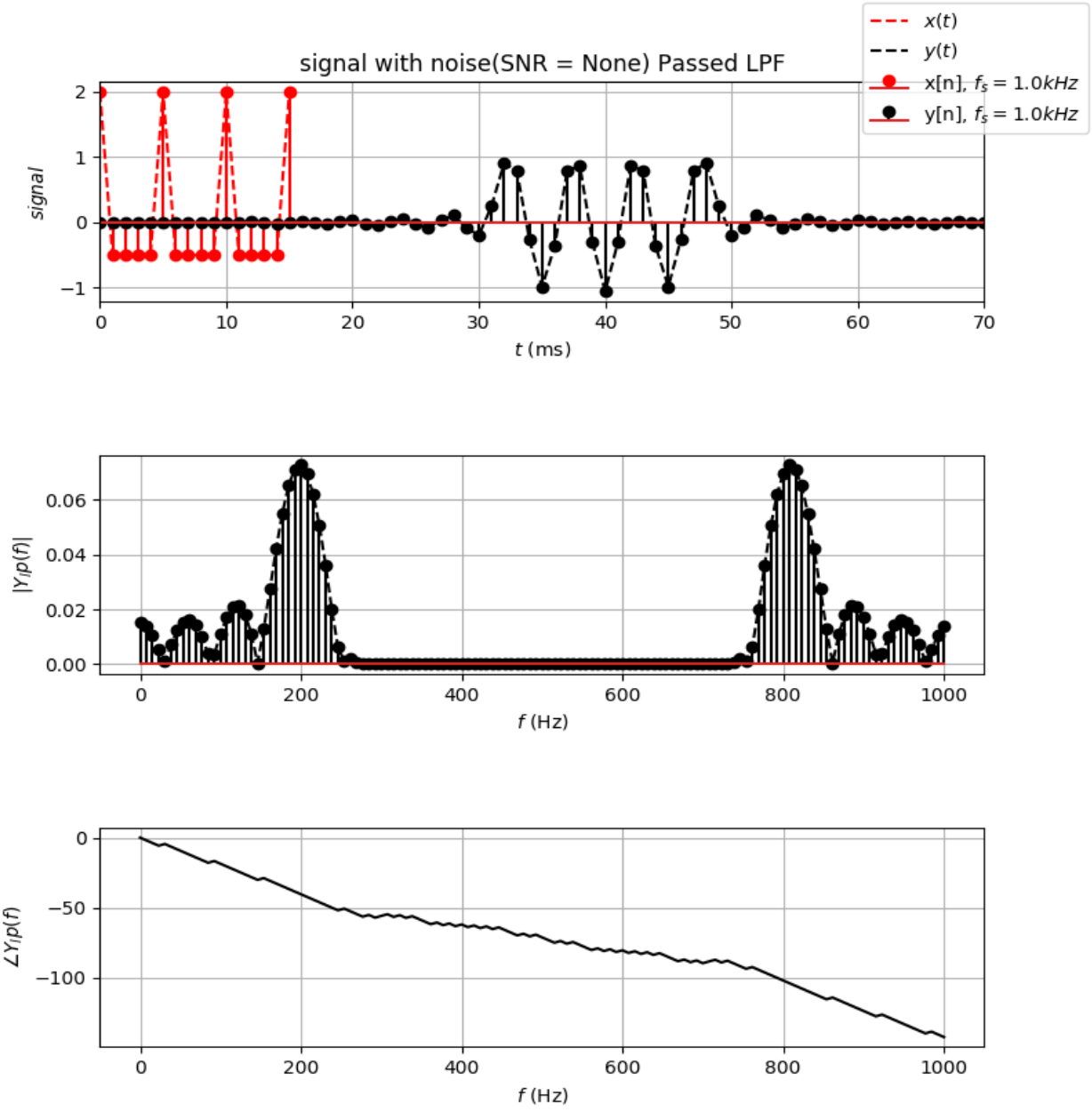
BPF 의 설계는 간단하였다. 이미 만들어놓은 LPF 클래스로 low LPF 와 high LPF 를 만들어 high_LPF - low_LPF 를 하면 되는 문제이기 때문이다. (LPF 에 - 붙으면 HPF 가 된다) 실험을 진행하면서 BPF 의 high 쪽 frequency 가 Nyquist rate $\frac{f_s}{2}$ 를 넘지 않도록 하는 것을 신경 썼다. (aliasing 방지)

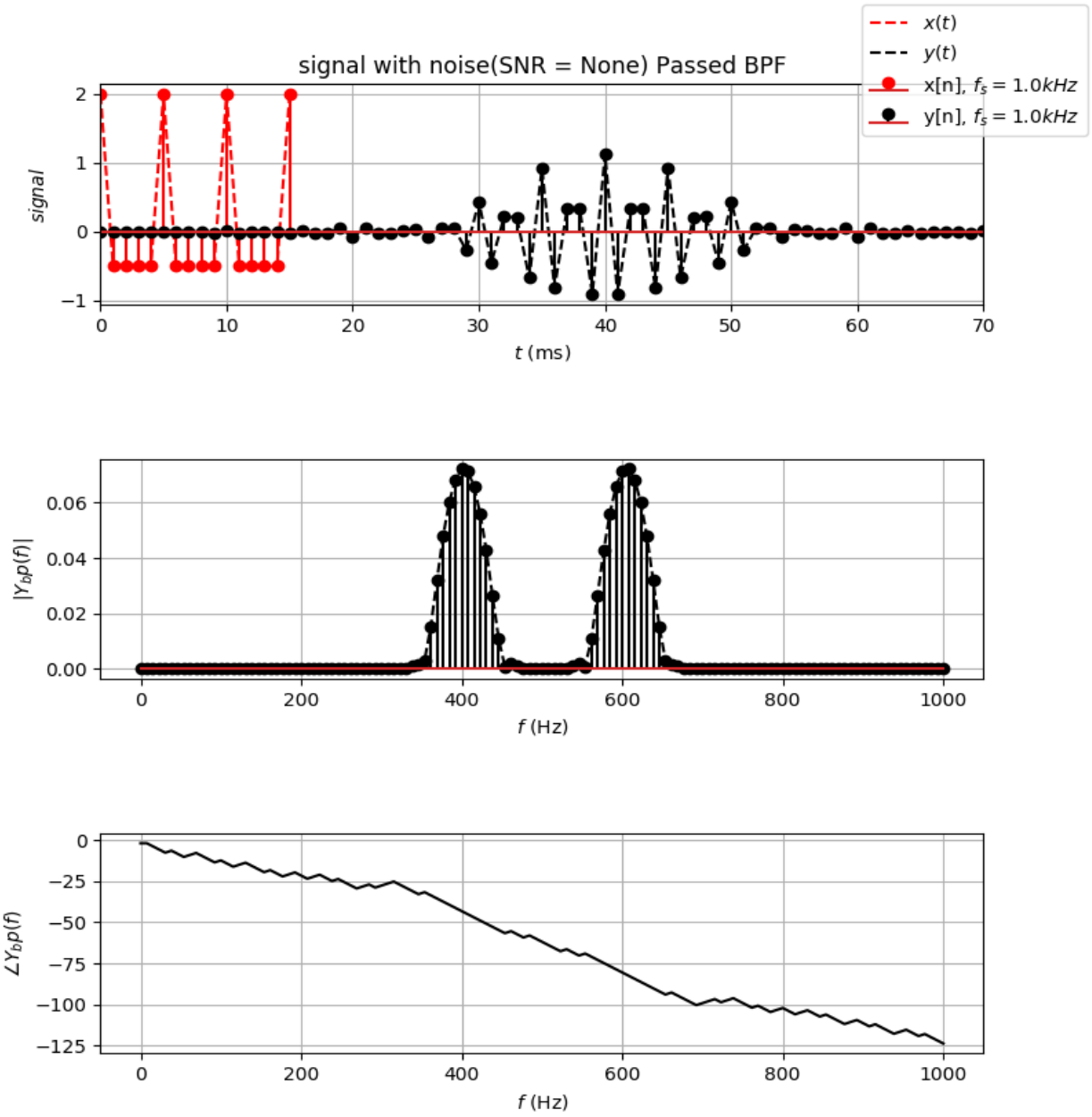


LPF 와 마찬가지로 $\theta(w) = -\alpha w T_s$ 인 linear phase response 특성을 확인 하였다. 하지만 BPF 는 high pass 기능 을 하는 -low LPF 쪽이 문제가 있었는데 기존에 설계하려는 스펙에 맞지 않게 low passband frequency 전에 더 깎이는 것이 확인되었다. 하지만 이는 N 의 문제임을 확인하였고 sampling frequency 를 높여 filter length N 을 늘리면 문제는 해결이 되었다. (참조 : BPF)

4. LPF 및 BPF 의 출력신호의 파형과 스펙트럼 특성

Noise 가 없는 signal 을 통과 시켰을 때 출력 신호를 확인하면 위에서 언급한 linear phase response 및, filter 의 기능이 문제 없음을 확인 할 수 있다.

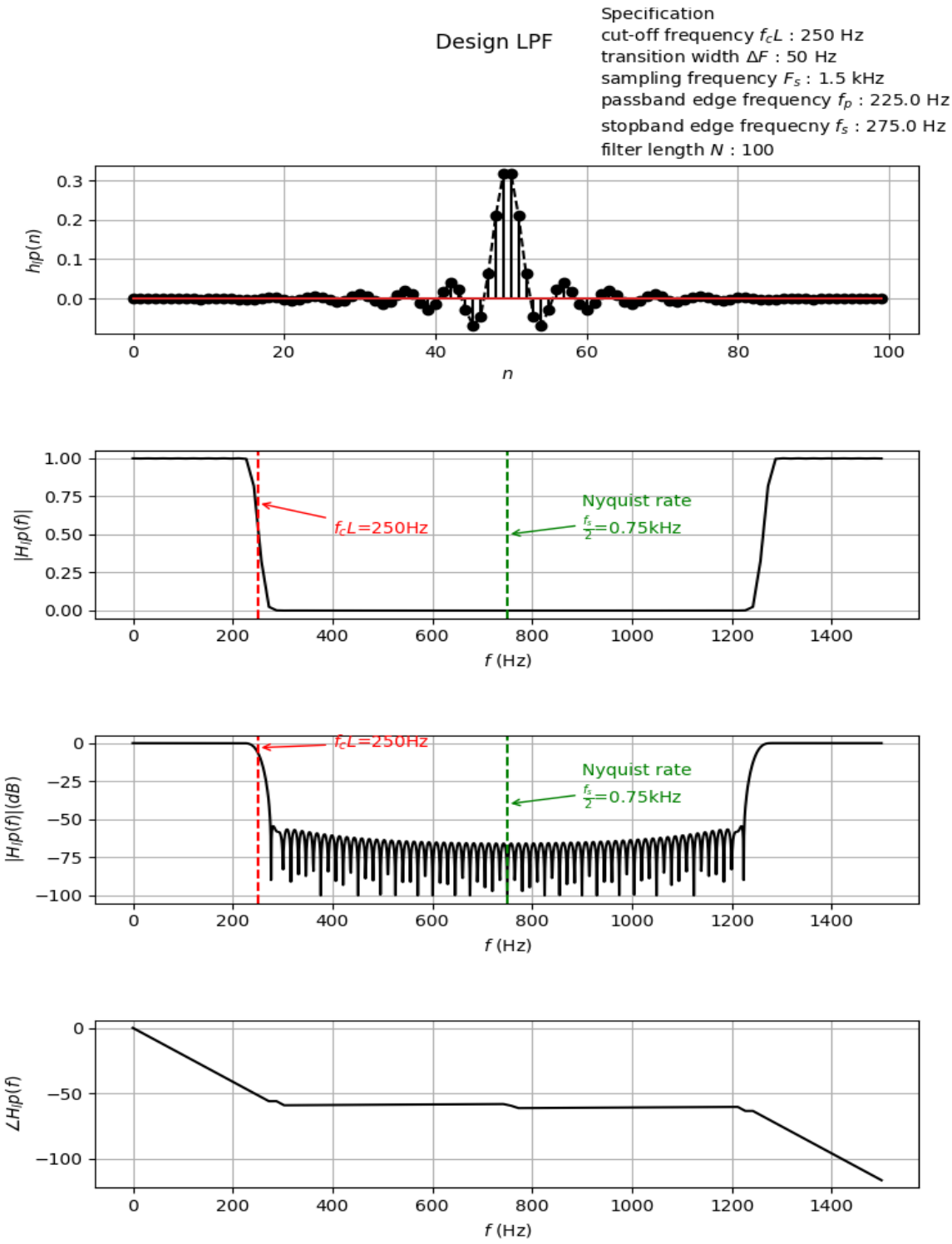




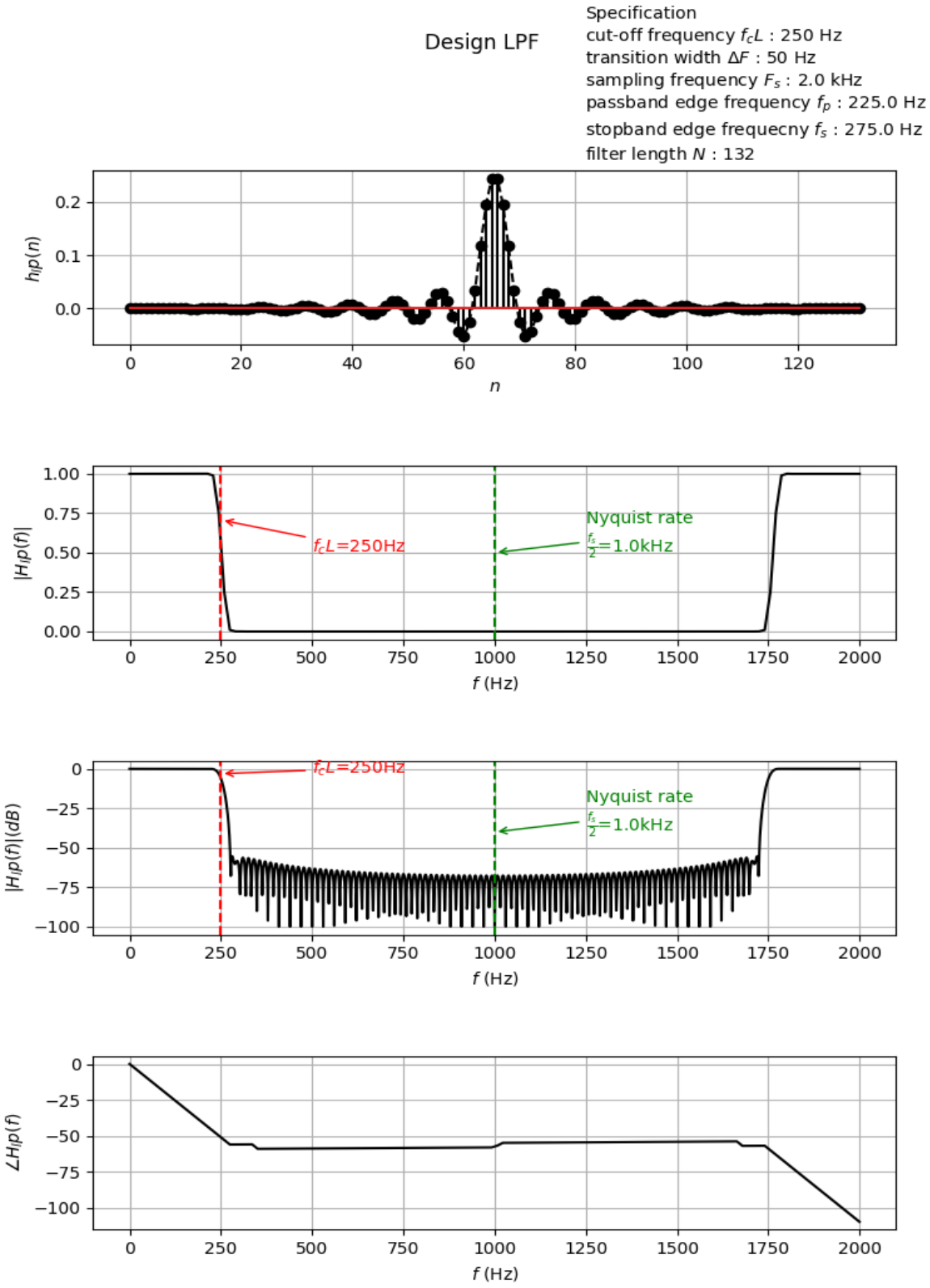
5. 표본 주파수를 변경하였을 경우에 설계된 LPF 의 사양(Spec.)의 특성 변화

코드를 진행 할 때 transition width 와 cut-off frequency 는 고정으로 하고 표본 주파수만을 변경하였다. 표본 주파수가 커질 수록 filter length N 이 커지는 것을 확인 할 수 있었다. Transition width 는 고정이므로 passband 이후 깎이는 정도는 같지만 스펙트럼을 보면 N 이 커질 수록 조금 더 매끄럽게(ideal 하게) 깎이고 또한 stopband 쪽 ripple 이 많이 줄어들고 attenuation 이 커지는 것을 확인 할 수 있다. 즉 표본주파수가 높아질 수록 필터가 점점 더 ideal 한 특성을 가진다.

1. $f_s = 1.5kHz$


Design LPF
Specification
cut-off frequency f_cL : 250 Hz
transition width ΔF : 50 Hz
sampling frequency F_s : 1.5 kHz
passband edge frequency f_p : 225.0 Hz
stopband edge frequecny f_s : 275.0 Hz
filter length N : 100
$h_{lp}(n)$
n
$|H_{lp}(f)|$
f_cL=250Hz
Nyquist rate
$\frac{f_s}{2}$=0.75kHz
f (Hz)
$|H_{lp}(f)|(dB)$
$\angle H_{lp}(f)$

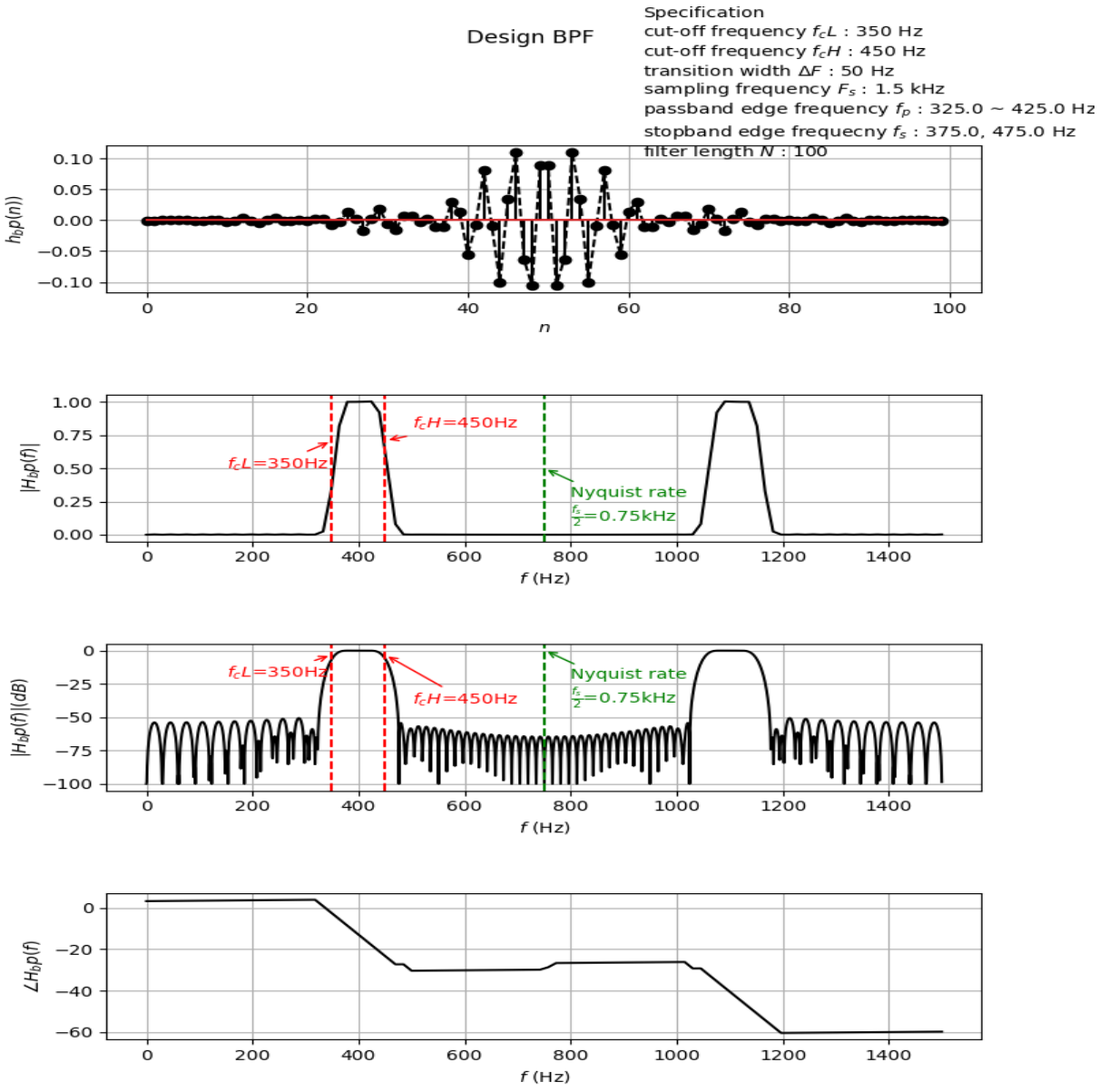
2. $f_s = 2kHz$


Design LPF
Specification
cut-off frequency f_cL : 250 Hz
transition width ΔF : 50 Hz
sampling frequency F_s : 2.0 kHz
passband edge frequency f_p : 225.0 Hz
stopband edge frequecny f_s : 275.0 Hz
filter length N : 132
$h_{lp}(n)$
n
$|H_{lp}(f)|$
f_cL=250Hz
Nyquist rate
$\frac{f_s}{2}$=1.0kHz
f (Hz)
$|H_{lp}(f)|(dB)$
$\angle H_{lp}(f)$

6. 결론

프로젝트를 진행하면서 다시 한번 DFT, FFT 의 개념을 확실히 이해 할 수 있었고, Filter 설계에 대해서도 알 수 있었다. 특히 X[k]와 X(w)의 관계, k 와 f 변환 및 필터의 특성 등 이론으로만 알았던 것들을 실제로 구현해보고 확인해보니 좋았다. 특히 matlab 이 아닌 python 으로 모든 것을 진행하면서 좋았던 점은 사용법이 매우 쉽고 무료라는 점, 특히 대규모 커뮤니티로 인해 많은 정보가 있다는 것이었다. 이미 만들어진 이런 DSP 관련 라이브러리나 함수를 통해 빠르고 쉽게 만들고 결과를 확인해보는 능력을 키운 것 같아 이후 다른 DSP 관련 프로젝트에 있어서 큰 도움이 될 것 같다.

7. 참조



Design BPF

Specification
cut-off frequency f_cL : 350 Hz
cut-off frequency f_cH : 450 Hz
transition width ΔF : 50 Hz
sampling frequency F_s : 2.0 kHz
passband edge frequency f_p : 325.0 ~ 425.0 Hz
stopband edge frequecny f_s : 375.0, 475.0 Hz
filter length N : 132