How to design N-Order Butterworth LPF

Innova Lee(이상훈)
gcccompil3r@gmail.com

s 를 jw 로 바꾸거나 jw 를 s 로 바꿔서 사용함

$$|H(j\omega)|^2 = T(s)T(-s)\Big|_{s=j\omega}$$

전달함수의 제곱 함수 $|T(j\omega)|^2 = |T(-j\omega)|^2$ 는 우함수다.
이 제곱 함수를 다항식으로 표현하면 분자와 분모 다항식이 모두 우함수로 구성되어야 한다.
이를 아래와 같이 표현하도록 한다.

$$|H(j\omega)|^2 = \frac{A(\omega^2)}{B(\omega^2)}$$

분자 부분을 A_0 라는 상수가 되도록 단순한 형식을 선택하도록 한다.

$$|H(j\omega)|^2 = \frac{A(\omega^2)}{B(\omega^2)} = \frac{A_0}{B_0 + B_2\omega^2 + B_4\omega^4 + B_6\omega^6 + \cdots + B_{2n}\omega^{2n}}$$

이와 같은 선택의 이유는 A 의 차수와 B 의 차수를 가능한 한 크게함으로써
수행되는 큰 ω 에 대해 $|H(j\omega)|$ 의 rolloff(감쇠되는 정도)를 크게 만들고 싶기 때문임
이 방식은 n-pole rolloff 가 있는 $|H(j\omega)|$ 와 all-pole 기능을 수행하는 $H(s)$ 를 제공한다.
B_0 과 B_{2n} 을 제외한 모든 B 계수가 0 이고, $A_0 = B_0$ 인 특수한 경우 $H(j0) = 1$ 이고 B_{2n} 은 아래와 같이 결정된다.

$$B_{2n} = \left(\frac{1}{\omega_0}\right)^{2n}$$

위 내용을 적용하여 전달함수를 아래와 같이 적어볼 수 있다.

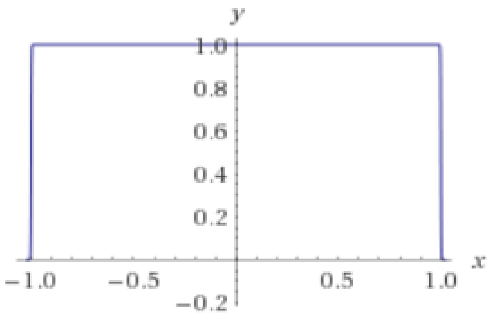
$$|H(j\omega)|^2 = \frac{A(\omega^2)}{B(\omega^2)} = \frac{1}{1 + \left(\frac{\omega}{\omega_0}\right)^{2n}}$$

1/(1 + x^1000)

Input:

$$\frac{1}{1+x^{1000}}$$

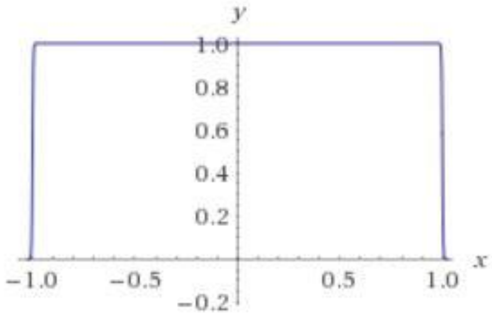
Plots:



Input:

$$\frac{1}{1+x^{500}}$$

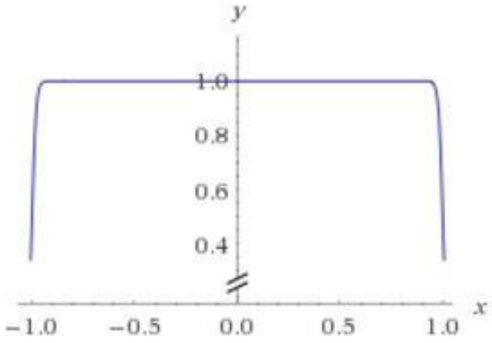
Plots:



Input:

$$\frac{1}{1+x^{100}}$$

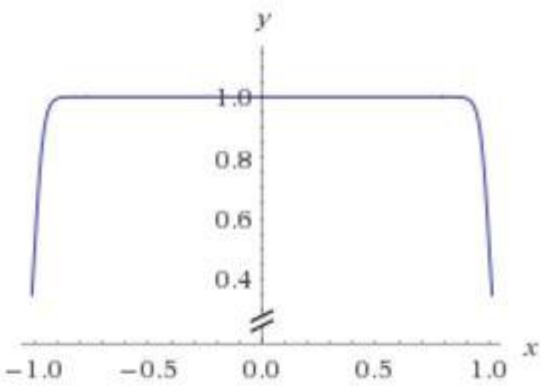
Plots:



Input:

$$\frac{1}{1 + x^{50}}$$

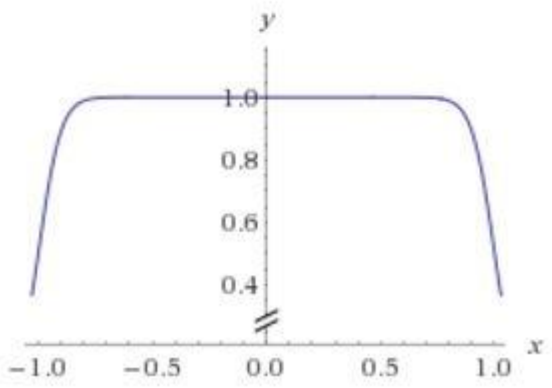
Plots:



Input:

$$\frac{1}{1 + x^{20}}$$

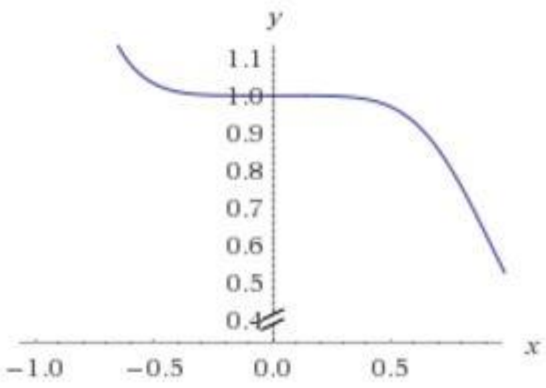
Plots:



Input:

$$\frac{1}{1 + x^{5}}$$

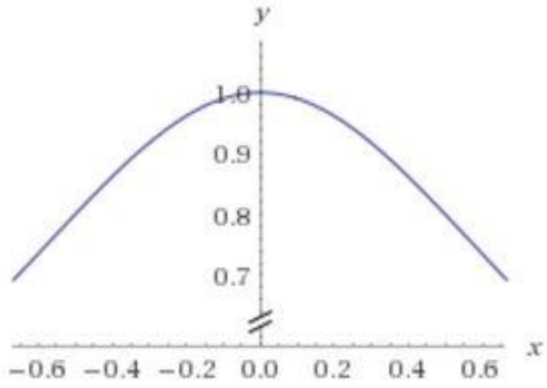
Plots:

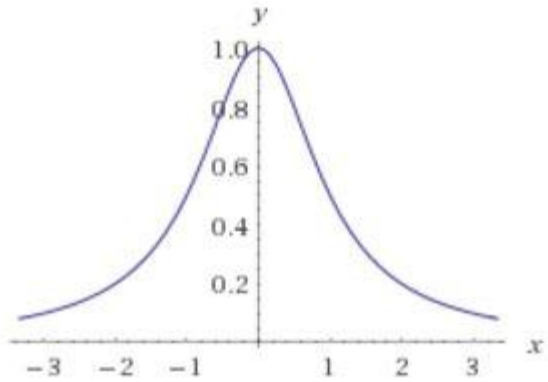


Input:

$$\frac{1}{1+x^2}$$

Plots:

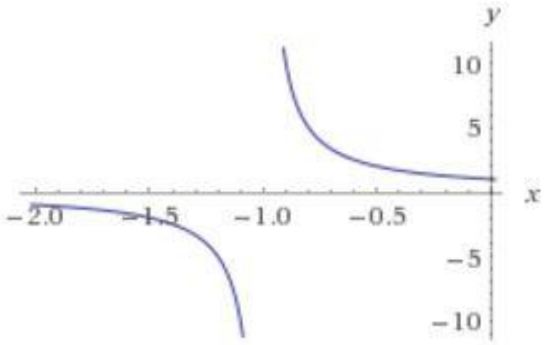


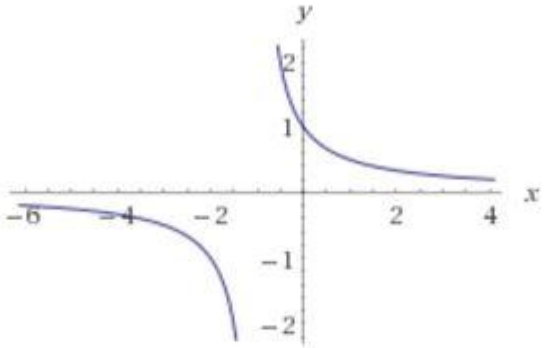


Input:

$$\frac{1}{1+x}$$

Plots:





$$H(s)H(-s) = |H(j\omega)|^2\Big|_{s=j\omega} = \frac{1}{1+\left(\frac{\omega}{\omega_0}\right)^{2N}}\Bigg|_{s=j\omega} = \frac{1}{1+\left(\frac{-s^2}{\omega_0^2}\right)^{N}}$$

혼동이 오니 ω_0 를 ω_c 로 표기하도록 함

$$\therefore |H(\omega)|^2 = \frac{1}{1+\left(\frac{-s^2}{\omega_c^2}\right)^{N}}$$

계수값은 분모가 0 이 되는 극점을 찾아보자!

$$\left(\frac{-s^2}{\omega_c^2}\right)^{N} = -1 = e^{j(2k-1)\pi}, \qquad k = 1, 2, \dots, N$$

$$\frac{-s^2}{\omega_c^2} = e^{\frac{j(2k-1)}{N}\pi}$$

$$s_k^2 = -\omega_c^2 e^{\frac{j(2k-1)}{N}\pi}, \qquad k = 1, 2, \dots, N$$

$$\therefore s_k = \pm j\omega_c e^{\frac{j(2k-1)}{2N}\pi}, \qquad k = 1, 2, \dots, N$$

단위원에서 j 가 있으므로 위상이 90 도 이동하였고 거기서 180 도는 전부 실수값이 음수에 해당한다.
그리고 나머지는 전부 오른쪽(양수)에 해당하므로 음수를 제거하면서 360 도가 아닌 위상의 절반만 활용해도됨

$$s_k = j\omega_c e^{\frac{j(2k-1)}{2N}\pi}, \qquad k = 1, 2, \dots, N$$

$$= j\omega_c\left\{cos\left(\frac{2k-1}{2N}\pi\right) + jsin\left(\frac{2k-1}{2N}\pi\right)\right\} = \left\{jcos\left(\frac{2k-1}{2N}\pi\right) - sin\left(\frac{2k-1}{2N}\pi\right)\right\}\omega_c$$

$$\left(\because sin\left(\frac{2k-1}{2N}\pi\right) \Rightarrow sin\left(\frac{1}{2N}\pi\right), sin\left(\frac{3}{2N}\pi\right), sin\left(\frac{5}{2N}\pi\right), \dots, sin\left(\frac{4N-2}{2N}\right)\pi\right)$$

$$H(s)H(-s) = |H(j\omega)|^2\Big|_{s=j\omega} = \frac{1}{1+\left(\frac{\omega}{\omega_0}\right)^{2N}}\Bigg|_{s=j\omega} = \frac{1}{1+\left(\frac{-s^2}{\omega_0^2}\right)^{N}}$$

$$\therefore H(s) = \frac{1}{1+\left(\frac{s}{\omega_c}\right)^{N}} = \frac{1}{\prod_{k=1}^{N}\left(\frac{s}{\omega_c}-s_k\right)}, \qquad s_k = \left\{-sin\left(\frac{2k-1}{2N}\pi\right)+jcos\left(\frac{2k-1}{2N}\pi\right)\right\}$$

실제 각 N 에 대해 풀어본다면 아래와 같다.

$N = 1$

$$H(s) = \frac{1}{\frac{s}{\omega_c}-s_1}, \qquad s_1 = -sin\left(\frac{1}{2}\pi\right)+jcos\left(\frac{1}{2}\pi\right) = -1+0j = -1$$

$$\therefore H(s) = \frac{1}{\frac{s}{\omega_c}+1} = \frac{\omega_c}{s+\omega_c}$$

$N = 2$

$$H(s) = \frac{1}{\prod_{k=1}^{2}\left(\frac{s}{\omega_c}-s_k\right)} = \frac{1}{\left(\frac{s}{\omega_c}-s_1\right)\left(\frac{s}{\omega_c}-s_2\right)}$$

$$s_1 = -sin\left(\frac{1}{4}\pi\right)+jcos\left(\frac{1}{4}\pi\right) = -\frac{\sqrt{2}}{2}+j\frac{\sqrt{2}}{2}, \qquad s_2 = -sin\left(\frac{3}{4}\pi\right)+jcos\left(\frac{3}{4}\pi\right) = -\frac{\sqrt{2}}{2}-j\frac{\sqrt{2}}{2}$$

$$\therefore H(s) = \frac{1}{\left(\frac{s}{\omega_c}+\frac{\sqrt{2}}{2}-j\frac{\sqrt{2}}{2}\right)\left(\frac{s}{\omega_c}+\frac{\sqrt{2}}{2}+j\frac{\sqrt{2}}{2}\right)} = \frac{{\omega_c}^2}{s^2+\sqrt{2}s\omega_c+\omega_c^2} = \frac{1}{\frac{s^2}{\omega_c^2}+\sqrt{2}\frac{s}{\omega_c}+1}$$

$N = 3$ **일종의 트릭으로** $\omega_c = 1$ **로 지정하고 나중에** $s = s/\omega_c$ **로 치환하면됨**

$$H(s)\Big|_{\omega_c=1} = \frac{1}{\prod_{k=1}^{N}(s-s_k)} = \frac{1}{(s-s_1)(s-s_2)(s-s_3)}, \qquad s_1 = -sin\left(\frac{1}{6}\pi\right)+jcos\left(\frac{1}{6}\pi\right) = -\frac{1}{2}+j\frac{\sqrt{3}}{2}$$

$$s_2 = -sin\left(\frac{3}{6}\pi\right)+jcos\left(\frac{3}{6}\pi\right) = -1+0j, \qquad s_3 = -sin\left(\frac{5}{6}\pi\right)+jcos\left(\frac{5}{6}\pi\right) = -\frac{1}{2}-j\frac{\sqrt{3}}{2}$$

$$H(s) = \frac{1}{(s+1)\left(s+\frac{1}{2}-j\frac{\sqrt{3}}{2}\right)\left(s+\frac{1}{2}+j\frac{\sqrt{3}}{2}\right)} = \frac{1}{(s+1)\left\{\left(s+\frac{1}{2}\right)^2+\frac{3}{4}\right\}} = \frac{1}{(s+1)(s^2+s+1)}$$

$$\therefore H(s) = \frac{1}{s^3+2s^2+2s+1} = \frac{1}{\frac{s^3}{\omega_c^3}+2\frac{s^2}{\omega_c^2}+2\frac{s}{\omega_c}+1}$$

먼저 앞서 구했던 값들이 Butterworth Filter 에 Table 에 대한 값과 동일함을 알 수 있다.

Denominator coefficients for polynomials of the form $S_n + a_{n-1}s^{n-1} + a_{n-2}s^{n-2} + \ldots\ldots\ldots\ldots + a_1 s + a_0$

n	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9
1	1									
2	1	1.414								
3	1	2.000	2.000							
4	1	2.613	3.414	2.613						
5	1	3.236	5.236	5.236	3.236					
6	1	3.864	7.464	9.142	7.464	3.864				
7	1	4.494	10.098	14.592	14.592	10.098	4.494			
8	1	5.126	13.137	21.846	25.688	21.846	13.137	5.126		
9	1	5.759	16.582	31.163	41.986	41.986	31.163	16.582	5.759	
10	1	6.392	20.432	42.802	64.882	74.233	64.882	42.802	20.432	6.392

인터넷 상에는 10차까지만 나와 있는데 그 이상이 필요하다면
지금 이 기법으로 20차 혹은 그 이상도 직접 설계하여 적용할 수 있을 것이다.

이제 본격적으로 차단 주파수가 주어졌을때 필터 계수 값을 구해보도록 하자!
2 차 Butterworth LPF 와 3 차 Butterworth LPF 를 모두 구해보자!
우선 정규화된 디지털 필터의 차단 주파수는 아래와 같다.

$$\omega_c = \frac{2\pi f_c}{f_s} = \frac{2\pi(800MHz)}{12GHz} = 0.4189$$

실제 회로상의 아날로그 차단 주파수로 변환한다.

$$s = \frac{z-1}{z+1} \equiv \frac{1-z^{-1}}{1+z^{-1}}$$

$$\omega_{ac} = tan\left(\frac{\omega_c}{2}\right) = 0.2126$$

$$H(s) = \frac{1}{\frac{s^2}{\omega_c^2} + \sqrt{2}\frac{s}{\omega_c} + 1} = \frac{1}{\frac{s^2}{0.2126^2} + \sqrt{2}\frac{s}{0.2126} + 1}$$

$$H(s) = \frac{1}{\frac{1}{0.2126^2}\left[\frac{1-z^{-1}}{1+z^{-1}}\right]^2 + \frac{\sqrt{2}}{0.2126}\left[\frac{1-z^{-1}}{1+z^{-1}}\right] + 1}$$

$$\frac{1}{0.2126^2} = 22.1245, \quad \frac{\sqrt{2}}{0.2126} = 6.6520$$

$$\frac{42.2490}{29.7765} = 1.4189, \quad \frac{16.4725}{29.7765} = 0.5532$$

$$\frac{1}{29.7765} = 0.0336, \quad \frac{2}{29.7765} = 0.0672$$

정리해서 계산만 수행하면 완료된다.
이 작업도 컴퓨터 시켜도 무방하지만 for 문 돌면서
소요되는 몇 clock 이라도 아끼고자 한다면 반드시 직접 계산해서 수치값으로 코딩하는 것이 좋음

$$H(s) = \frac{1}{\frac{1}{0.2126^2}\left[\frac{1-2z^{-1}+z^{-2}}{1+2z^{-1}+z^{-2}}\right] + \frac{\sqrt{2}}{0.2126}\left[\frac{1-z^{-1}}{1+z^{-1}}\right] + 1} = \frac{1}{22.1245\left[\frac{1-2z^{-1}+z^{-2}}{1+2z^{-1}+z^{-2}}\right] + 6.6520\left[\frac{1-z^{-1}}{1+z^{-1}}\right]\left[\frac{1+z^{-1}}{1+z^{-1}}\right] + 1}$$

$$= \frac{1}{\frac{22.1245(1-2z^{-1}+z^{-2}) + 6.6520(1-z^{-2}) + 1 + 2z^{-1} + z^{-2}}{1+2z^{-1}+z^{-2}}}$$

$$= \frac{1+2z^{-1}+z^{-2}}{29.7765 - 42.2490z^{-1} + 16.4725z^{-2}} = \frac{0.0336 + 0.0672z^{-1} + 0.0336z^{-2}}{1 - 1.4189z^{-1} + 0.5532z^{-2}}$$

최종적으로 Z-Transform 한 결과를 이산화 시켜서 디지털 필터의 설계를 마무리한다.
(라플라스 변환의 디지털 기법이 Z-Transform 이라고 보면됨)

$$\boldsymbol{H(s)=\frac{1+2z^{-1}+z^{-2}}{29.7765-42.2490z^{-1}+16.4725z^{-2}}=\frac{0.0336+0.0672z^{-1}+0.0336z^{-2}}{1-1.4189z^{-1}+0.5532z^{-2}}}$$

항상 기억해야할 것은 전달함수는 뭔가 입력과 출력의 비율이였음을 상기하면됨

$$\boldsymbol{H(s)=\frac{Y(z)}{X(z)}=\frac{0.0336+0.0672z^{-1}+0.0336z^{-2}}{1-1.4189z^{-1}+0.5532z^{-2}}}$$
$$\boldsymbol{=\left(1-1.4189z^{-1}+0.5532z^{-2}\right)Y(z)=\left(0.0336+0.0672z^{-1}+0.0336z^{-2}\right)X(z)}$$
$$\boldsymbol{\therefore y_i=1.4189y_{i-1}-0.5532y_{i-2}+0.0336x_i+0.0672x_{i-1}+0.0336x_{i-2}}$$

이것을 프로그래밍해서 구현하면 아주 스펙트럼 분석시 아주 깔끔하게 남김없이 제거하는 것을 볼 수 있다.
차수가 높으면 높을수록 정교하게 작업을 수행할 수 있다.
실시간으로 수행하기 위해선 샘플링한 신호를 CPU, DSP 혹은 기타 보조 연산 장치들이
병렬로 연산하는 속도가 샘플링하는 속도보다 빨라야 정상적으로 Real-Time 분석이 가능하다.
그렇기 때문에 고속 ADC 도 중요하고 또한 프로그램의 최적화 정도와 캐시의 활용도도 중요해짐
한가지 아쉬운 점이라면 CPU 의 Pipeline 은 Branch Instruction 을 만나게되면 깨지게되어 있음
(Branch Prediction 을 수행할 수 있는 CPU 라면 이 문제를 보다 최소화시킬 수 있음)

그렇다고 전부 하드코딩으로 때려넣으면 샘플링하는 신호의 수가 많아질수록 노가다가 엄청나짐
뿐만 아니라 코드 크기가 증가하면서 일반적인 운영체제(리눅스)의 Page 크기에 해당하는
4K 공간을 넘어갈 수 있어 또 성능 저하를 유발할 수 있음

결국 적절하게 Trade-Off 를 봐야하는데 이런 경우 SIMD Instruction 을 활용하면 좋음
(보다 더 최적화 하기 위해선 Loop Unrolling 을 위한 적절한 하드코딩과 어셈블리어, for 문을 혼합하면 좋음)

이제 3 차 Butterworth LPF 의 계수를 구해보도록 하자!

$$\frac{1}{0.2126^3} = 104.0663, \quad \frac{2}{0.2126^2} = 44.2490$$
$$\frac{2}{0.2126} = 9.4073, \quad \frac{344.0406}{158.7226} = 2.1676$$
$$\frac{261.5426}{158.7226} = 1.6478, \quad \frac{68.2246}{158.7226} = 0.4298$$
$$\frac{1}{158.7226} = 0.0063, \quad \frac{3}{158.7226} = 0.0189$$

$$\omega_c = \frac{2\pi f_c}{f_s} = \frac{2\pi(800\ MHz)}{12\ GHz} = 0.4189, \qquad \omega_{ac} = tan\left(\frac{\omega_c}{2}\right) = 0.2126$$

$$H(s) = \frac{1}{\frac{s^3}{\omega_c^3} + 2\frac{s^2}{\omega_c^2} + 2\frac{s}{\omega_c} + 1} = \frac{1}{\frac{s^3}{0.2126^3} + 2\frac{s^2}{0.2126^2} + 2\frac{s}{0.2126} + 1}$$

$$= \frac{1}{\frac{1}{0.2126^3}\left[\frac{1-z^{-1}}{1+z^{-1}}\right]^3 + 2\frac{1}{0.2126^2}\left[\frac{1-z^{-1}}{1+z^{-1}}\right]^2 + 2\frac{1}{0.2126}\left[\frac{1-z^{-1}}{1+z^{-1}}\right] + 1}$$

$$= \frac{1}{\frac{1}{0.2126^3}\left[\frac{1-3z^{-1}+3z^{-2}-z^{-3}}{1+3z^{-1}+3z^{-2}+z^{-3}}\right] + \frac{2}{0.2126^2}\left[\frac{1-2z^{-1}+z^{-2}}{1+2z^{-1}+z^{-2}}\right] + \frac{2}{0.2126}\left[\frac{1-z^{-1}}{1+z^{-1}}\right] + 1}$$

$$= \frac{1}{\frac{1}{0.2126^3}\left[\frac{1-3z^{-1}+3z^{-2}-z^{-3}}{1+3z^{-1}+3z^{-2}+z^{-3}}\right] + \frac{2}{0.2126^2}\left[\frac{1-2z^{-1}+z^{-2}}{1+2z^{-1}+z^{-2}}\right]\left[\frac{1+z^{-1}}{1+z^{-1}}\right] + \frac{2}{0.2126}\left[\frac{1-z^{-1}}{1+z^{-1}}\right]\left[\frac{1+z^{-1}}{1+z^{-1}}\right]^2 + 1}$$

$$= \frac{1}{\frac{1}{0.2126^3}\left[\frac{1-3z^{-1}+3z^{-2}-z^{-3}}{1+3z^{-1}+3z^{-2}+z^{-3}}\right] + \frac{2}{0.2126^2}\left[\frac{1-z^{-1}-z^{-2}+z^{-3}}{1+3z^{-1}+3z^{-2}+z^{-3}}\right] + \frac{2}{0.2126}\left[\frac{1+z^{-1}+z^{-2}-z^{-3}}{1+3z^{-1}+3z^{-2}+z^{-3}}\right] + 1}$$

$$= \frac{1}{104.0663\left[\frac{1-3z^{-1}+3z^{-2}-z^{-3}}{1+3z^{-1}+3z^{-2}+z^{-3}}\right] + 44.2490\left[\frac{1-z^{-1}-z^{-2}+z^{-3}}{1+3z^{-1}+3z^{-2}+z^{-3}}\right] + 9.4073\left[\frac{1+z^{-1}+z^{-2}-z^{-3}}{1+3z^{-1}+3z^{-2}+z^{-3}}\right] + 1}$$

$$= \frac{1+3z^{-1}+3z^{-2}+z^{-3}}{104.0663(1-3z^{-1}+3z^{-2}-z^{-3}) + 44.2490(1-z^{-1}-z^{-2}+z^{-3}) + 9.4073(1+z^{-1}+z^{-2}-z^{-3}) + 1+3z^{-1}+3z^{-2}+z^{-3}}$$

$$= \frac{1+3z^{-1}+3z^{-2}+z^{-3}}{158.7226 - 344.0406z^{-1} + 261.5426z^{-2} - 68.2246z^{-3}}$$

$$= \frac{0.0063 + 0.0189z^{-1} + 0.0189z^{-2} + 0.0063z^{-3}}{1 - 2.1676z^{-1} + 1.6478z^{-2} - 0.4298z^{-3}} = \frac{Y(z)}{X(z)}$$

$$\left(1 - 2.1676z^{-1} + 1.6478z^{-2} - 0.4298z^{-3}\right)Y(z) = \left(0.0063 + 0.0189z^{-1} + 0.0189z^{-2} + 0.0063z^{-3}\right)X(z)$$

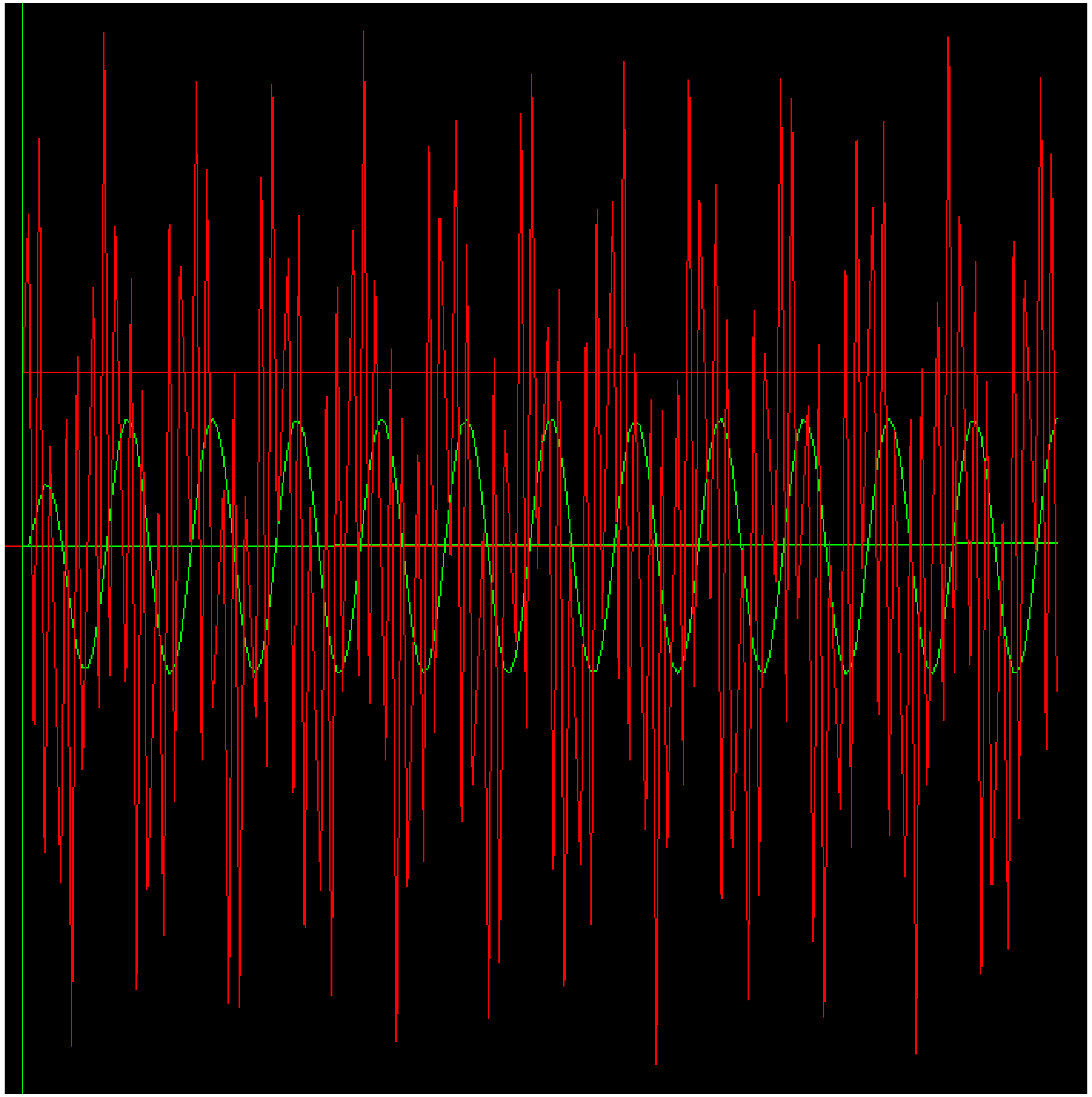
$$\therefore y_i = 2.1676y_{i-1} - 1.6478y_{i-2} + 0.4298y_{i-3} + 0.0063x_i + 0.0189x_{i-1} + 0.0189x_{i-2} + 0.0063x_{i-3}$$

동일한 방법으로 4 차, 5 차, N 차를 구할 수 있다.
사람이 직접 손으로 계산하기가 매우 번거로우므로 필터 계수를 구하는 코드를 작성하면 보다 편리하다.
그 외에 Matlab 을 통해서도 계쑤를 구할 수 있지만 사용할 수 없는 경우를 반드시 고려해야 한다.

C Based Implementation Results

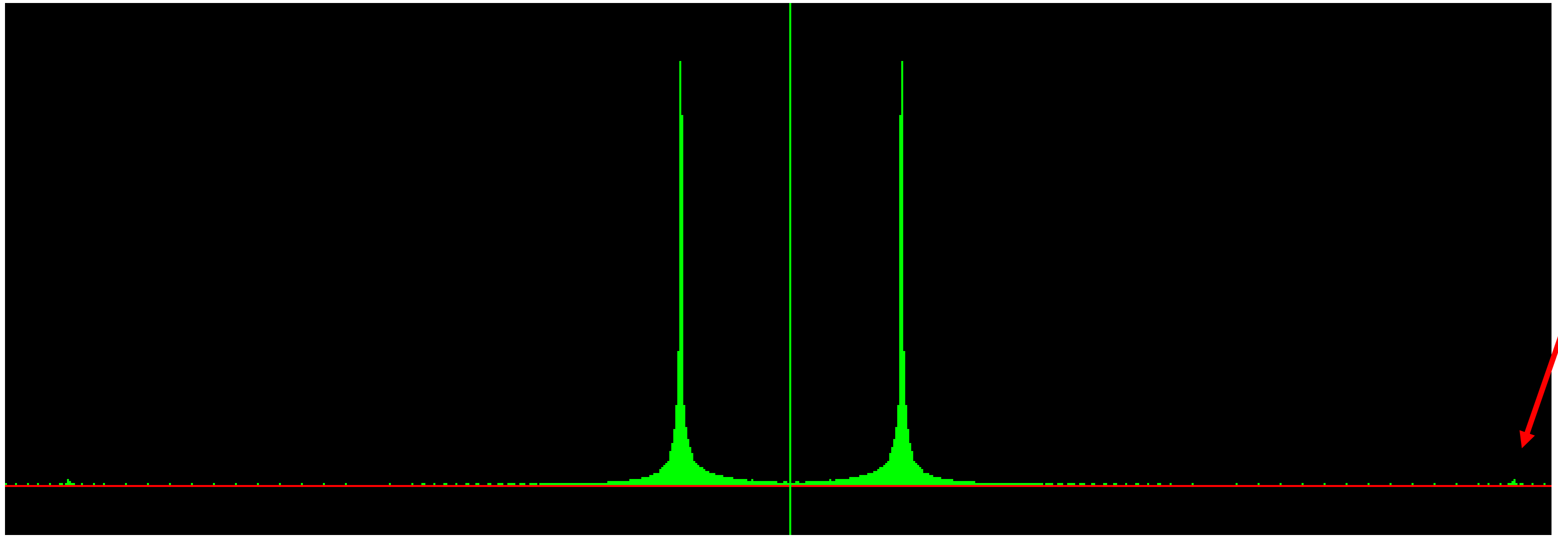
그래픽 처리는 OpenGL 로 작업하였다.

RED: Non-Filter
GREEN: Filter



C Based FFT Results

Perfect!



매우 만족스럽고도 완벽한 성분 제거를 수행했음을 볼 수 있다.

C Code Segments

```c
        glEnd();

        glColor3f(0.0, 1.0, 0.0);

        glBegin(GL_LINE_LOOP);
        glVertex3f(0.0, 100.0, 0.0);
        glVertex3f(0.0, -100.0, 0.0);
        glEnd();

        //draw_omega_sin();
        //draw_spectrum();
        //low_pass_filter(lpf_signal);
        order2_butterworth_filter(lpf_signal);
        spectrum_analysis(lpf_signal);
        glutSwapBuffers();
}

void reshape(int w, int h)
{
        GLfloat n_range = 100.0f;
```

```c
for(i = 2; i < SLICE; i++)
{
        lpf[i] = 1.4189 * lpf[i - 1] - 0.5532 * lpf[i - 2] +
                 0.0336 * signal[i] + 0.0672 * signal[i - 1] +
                 0.0336 * signal[i - 2];
       printf("lpf[%d] = %lf\n", i, lpf[i]);
}
```

Butterworth IIR LPF

FFT

Calculated Filter Coefficients