Tugas EB3102 - Pengolahan Sinyal Biomedika

Irfan Tito Kurniawan - 18317019

23 Desember 2019

1 Topik I: Analisis Sinyal

1.1 Teori

1.1.1 Electrocardiogram

Electrocardiogram, selanjutnya disebut dengan **ECG**, merupakan sinyal listrik yang merepresentasikan aktivitas jantung. ECG merupakan salah satu biosinyal utama yang digunakan untuk membantu proses diagnosis penyakit. Umumnya, sinyal ECG diteliti dalam *domain* waktu, sehingga dibutuhkan sinyal ECG yang bersih dan benar-benar merepresentasikan aktivitas jantung. Akan tetapi, pada kenyataannya, sinyal ECG yang memiliki amplitudo yang relatif kecil terbaca bercampur dengan derau yang kebanyakan memiliki amplitudo lebih besar dibandingkan dengan sinyal ECG. Karena itu, diperlukan proses *filtering* untuk memperoleh sinyal ECG yang bersih dan dapat diandalkan.

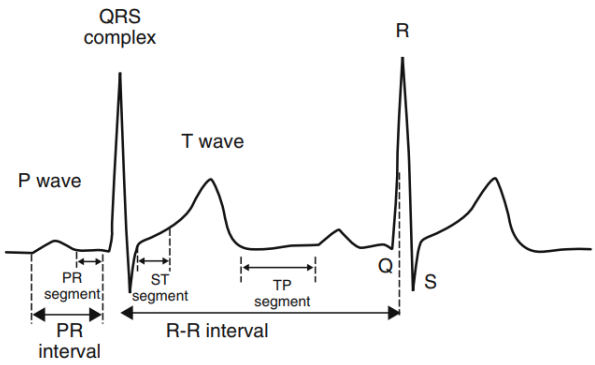

Figure 1: Sinyal ECG [ASSK14]

Sinyal ECG dapat dibagi menjadi beberapa segmen: gelombang P, kompleks QRS, serta gelombang T. Tiap segmen memiliki rentang frekuensi sinyal yang berbeda-beda. Dengan transformasi Fourier dari gelombang ECG, didapati bahwa segmen gelombang P-T secara umum memiliki rentang frekuensi sinyal antara 0.5 hingga 10 Hz sedangkan kompleks QRS memiliki rentang frekuensi sinyal antara 4 hingga 20 Hz [TWT84].

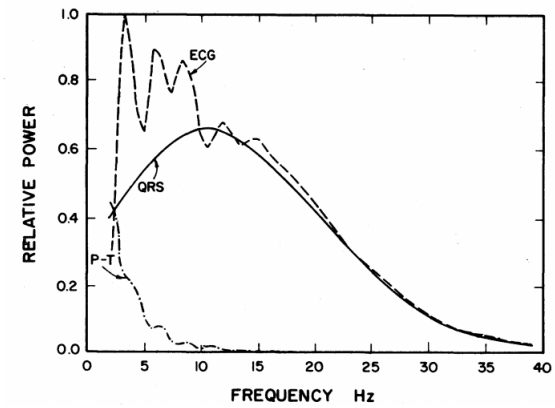

Figure 2: Spektrum Frekuensi Sinyal ECG [TWT84]

1.1.2 Derau

Derau yang mengotori sinyal ECG berasal dari berbagai sumber. Derau tersebut di antaranya: interferensi jala-jala, derau kontak elektroda, artefak akibat gerakan, kontraksi otot, dan pergeseran *baseline* atau dasar sinyal [ASSK14]. Tiap jenis derau memiliki rentang frekuensi yang berbeda-beda. Pada laporan ini, derau pada pembacaan sinyal ECG akan dibagi menjadi beberapa jenis, yakni derau akibat interferensi jala-jala, derau yang menyebabkan pergeseran *baseline*, serta derau berfrekuensi tinggi.

Interferensi jala-jala merupakan derau akibat adanya kapasitansi parasitik antara jala-jala dengan tubuh. Karenanya, derau akibat interferensi jala-jala memiliki frekuensi yang sama dengan frekuensi tegangan jala-jala beserta dengan harmonisanya. Sebagai contoh, apabila tegangan jala-jala pada tempat pengukuran sebesar 120 VAC 60 Hz, maka akan muncul derau akibat interferensi jala-jala pada frekuensi 60 Hz, 120 Hz, 180 Hz, begitu seterusnya. Derau jenis ini boleh jadi memiliki frekuensi yang *overlapping* dengan frekuensi informasi sinyal ECG. Untuk menghilangkan derau jenis ini, pada umumnya digunakan **Notch Filter** atau **Comb Filter**.

Derau yang menyebabkan pergeseran *baseline* merupakan derau berfrekuensi rendah, pada umumnya memiliki frekuensi DC atau 0 Hz hingga 0.8 Hz [ASSK14]. Karenanya, derau jenis ini boleh jadi pula memiliki frekuensi yang *overlapping* dengan frekuensi informasi sinyal ECG. Derau jenis ini dapat disebabkan oleh proses pernapasan dan gerakan subyek. Derau jenis ini dapat dihilangkan dengan **High-pass Filter** dengan frekuensi *cutoff* rendah.

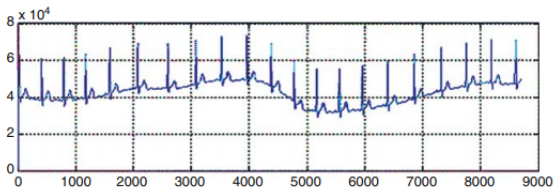

Figure 3: Sinyal ECG dengan Derau Penggeser *Baseline* [TWT84]

Kemudian, jenis derau yang ketiga adalah derau berfrekuensi tinggi. Derau jenis ini menyebabkan adanya *jitter* atau fluktuasi cepat sinyal ECG. Derau berfrekuensi tinggi dapat disebabkan oleh kontraksi otot dan derau akibat kontak elektroda yang kurang baik. Derau jenis ini memiliki rentang frekuensi yang sangat lebar, dari dalam rentang frekuensi sinyal ECG hingga di atas frekuensi sinyal ECG. Derau jenis ini dapat dimodelkan sebagai perubahan *baseline* yang terjadi secara cepat dan acak yang meluruh secara eksponensial [ASSK14]. Karena itu, selain dapat dihilangkan dengan **Low-pass Filter** dengan frekuensi *cutoff* tinggi, derau jenis ini juga akan hilang ketika bacaan sinyal dirata-ratakan.

1.1.3 Analisis Sinyal

Untuk menghilangkan derau dari sinyal ECG yang ada, pertama kita perlu mengetahui jenis derau yang ada dalam sinyal beserta dengan karakteristiknya. Salah satu cara paling mudah untuk mengetahui jenis dan karakteristik derau dalam sinyal adalah dengan melihat *Power Spectral Density* (PSD) dari sinyal yang diperoleh. PSD dari suatu sinyal menunjukkan bagaimana daya sinyal tersebut tersebar dalam *domain* frekuensi [OH18].

Spektrum daya pada frekuensi ω, $S_{xx}(\omega)$, dari suatu sinyal $x(n)$ yang memiliki transformasi Fourier $X(\omega)$ didefinisikan sebagai

$$S_{xx}(\omega) = |X(\omega)|^2 \tag{1}$$

Dengan demikian, PSD dari sinyal $x(n)$ dapat diperoleh dengan melakukan transformasi Fourier pada sinyal tersebut, kemudian mengkuadratkan magnituda spektrum sinyal pada *domain* frekuensi.

Dari persamaan diatas, diturunkan proses analisis sinyal berikut

Figure 4: Diagram Alir Analisis Sinyal

1.2 Hasil dan Pembahasan

Diberikan dua buah sampel sinyal ECG untuk dianalisis. Selanjutnya, sinyal dalam *file* 'psb_ecg1.dat' akan disebut sebagai 'sampel ECG 1' dan sinyal dalam *file* 'psb_ecg2.dat' akan disebut sebagai 'sampel ECG 2'. Kedua sinyal ini dianalisis dan ditampilkan sesuai dengan diagram alir pada bagian sebelumnya. Untuk menghitung transformasi Fourier dari kedua sinyal ini, digunakan algoritma *Fast Fourier Transform* (FFT), kemudian PSD dari sinyal dihitung dengan mengkuadratkan tiap spektrum hasil FFT dari sinyal.

Sinyal sampel ECG 1 akan dibahas terlebih dahulu. Berikut adalah plot sinyal sampel ECG 1 dalam *domain* waktu beserta dengan PSD-nya

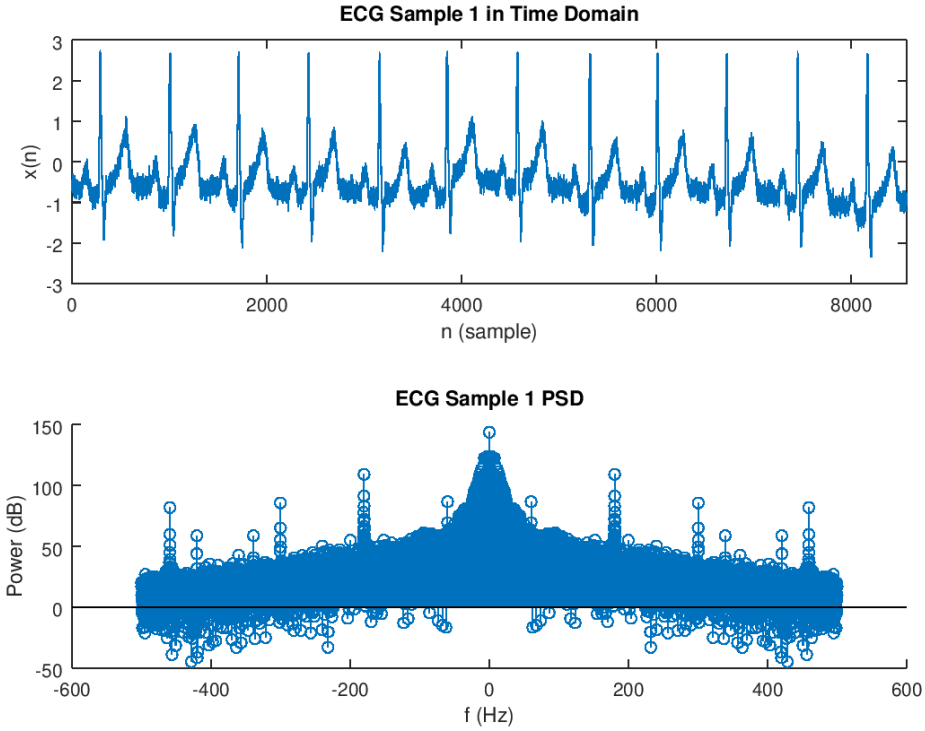

Figure 5: Plot sampel ECG 1 dalam *domain* waktu beserta PSD-nya

Sinyal sampel ECG 1 dalam *domain* waktu berbentuk menyerupai bentuk *template* dari sinyal ECG seperti pada gambar 1. Akan tetapi, bentuk sinyal sampel ECG 1 memiliki dua perbedaan utama dari bentuk *template* sinyal ECG, yakni sinyal bergerak naik turun secara perlahan (mengalami pergeseran *baseline*) yang menyebabkan letak puncak dan dasar sinyal berbeda antar siklus, serta terdapat fluktuasi sinyal yang terjadi secara cepat, atau selanjutnya disebut dengan *jitter*, yang menyebabkan sinyal tidak 'halus'.

Karena sinyal sampel ECG 1 dicuplik dengan *sample frequency* sebesar 1 kHz, akibat Teorema *Sampling* Nyquist-Shannon, hasil FFT dari sinyal sampel ECG 1 memiliki rentang frekuensi antara -500 Hz hingga 500 Hz. Karena sinyal sampel ECG 1 bersifat *real*, maka hasil FFT dari sinyal sampel ECG 1 bersifat genap. PSD dari sinyal merupakan kuadrat dari spektrum hasil FFT dari sinyal sampel ECG 1 sehingga PSD dari sinyal sampel ECG 1 pun bersifat genap dan memiliki rentang antara -500 Hz hingga 500 Hz.

Sinyal ECG yang bersih memiliki rentang frekuensi 0.5 hingga 20 Hz [TWT84]. Pada PSD dari sampel ECG 1, tampak bahwa sinyal memiliki komponen frekuensi dari 0 hingga 500 Hz, dengan puncak-puncak pada frekuensi kelipatan 60 Hz yang merupakan frekuensi jala-jala yang digunakan, dan pada 0 Hz. Hal ini menunjukkan bahwa ketiga jenis derau yang dijelaskan pada bagian sebelumnya mencampuri sinyal sampel ECG 1, yakni derau berfrekuensi rendah, derau akibat interferensi

jala-jala, serta derau berfrekuensi tinggi.

Derau berfrekuensi rendah pada sinyal sampel ECG 1 memiliki puncak pada frekuensi 0 Hz. Derau ini menyebabkan sinyal sampel ECG 1 mengalami pergeseran *baseline* sehingga sinyal bergerak naik turun perlahan dalam *domain* waktu.

Derau akibat interferensi jala-jala muncul pada frekuensi kelipatan 60 Hz. Akan tetapi, perhatikan bahwa derau hanya muncul pada kelipatan ganjil dari 60 Hz saja, yakni pada 180 Hz, 300 Hz, 420 Hz serta 840 Hz dan 960 Hz yang mengalami *frequency folding* dan masing-masing muncul pada frekuensi 340 Hz dan 460 Hz. Hal ini menunjukkan bahwa sinyal jala-jala memiliki sifat *half-wave symmetry* sehingga transformasi Fourier dari sinyal tersebut tidak memiliki harmonisa genap [GS12].

Kemudian, meski tidak memiliki puncak, pada PSD tampak bahwa di atas rentang frekuensi sinyal ECG, terdapat komponen sinyal dengan daya sekitar 100 dB pada frekuensi tepat di atas 20 Hz, hingga 30 dB pada frekuensi 500 Hz. Hal ini menunjukkan bahwa terdapat derau yang tersebar pada frekuensi di atas rentang frekuensi sinyal ECG. Derau ini, bersama dengan derau akibat interferensi jala-jala menyebabkan *jitter* pada sinyal sampel ECG 1 pada *domain* waktu.

Kemudian, berikut adalah plot sinyal sampel ECG 2 dalam *domain* waktu beserta dengan PSD-nya

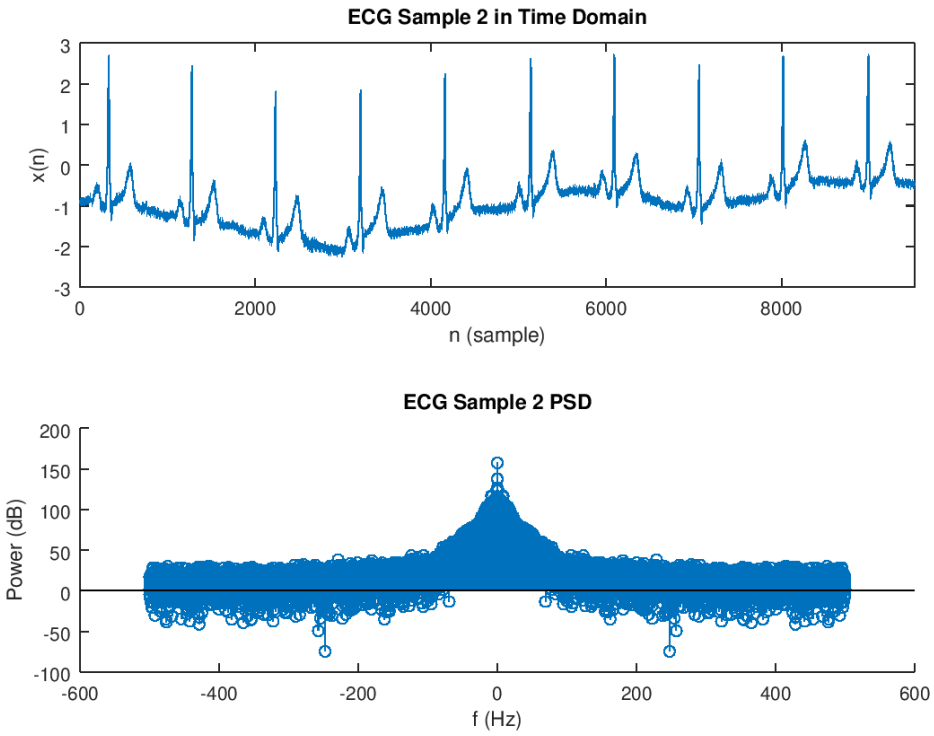

Figure 6: Plot sampel ECG 2 dalam *domain* waktu beserta PSD-nya

Hampir sama seperti sinyal sampel ECG 1, sinyal sampel ECG 2 pun memiliki bentuk umum

menyerupai bentuk *template* sinyal ECG. Hanya saja, sinyal sampel ECG 2 terlihat mengalami pergeseran *baseline* yang lebih ekstrem dibandingkan sinyal sampel ECG 1. Selain itu, sinyal sampel ECG 2 juga mengalami *jitter* yang menyebabkan sinyal sampel ECG 2 tidak tampak 'halus'.

PSD sinyal sampel ECG 2 menunjukkan bahwa meskipun sinyal memiliki derau yang tersebar pada seluruh rentang frekuensi, derau pada frekuensi rendah bersifat dominan, memiliki daya hingga 150 dB. Selain itu, derau akibat interferensi jala-jala tidak terlihat pada frekuensi kelipatan 60 Hz yang merupakan frekuensi jala-jala yang digunakan. Kemudian, seperti pada sinyal sampel ECG 1, sinyal sampel ECG 2 pun memliki derau berfrekuensi tinggi yang tersebar dari atas rentang frekuensi sinyal ECG hingga 500 Hz.

Derau berfrekuensi rendah yang memiliki amplitudo tinggi menyebabkan *baseline* sinyal sampel ECG 2 naik turun secara ekstrem. Karena derau pada sinyal ini lebih kuat dibandingkan derau berfrekuensi rendah pada sinyal sampel ECG 1, sinyal sampel ECG 2 mengalami pergeseran *baseline* yang lebih ekstrem dibandingkan sinyal sampel ECG 1. Derau ini memiliki puncak pada 0 Hz dan memiliki rentang frekuensi antara 0 Hz hingga sekitar 0.5 Hz.

Derau berfrekuensi tinggi pada sinyal ini tersebar dari atas rentang frekuensi ECG hingga frekuensi maksimum sinyal, 500 Hz. Seperti pada sinyal sampel ECG 1, derau berfrekuensi tinggi menyebabkan *jitter* pada sinyal sampel ECG 2 dalam *domain* waktu, meskipun *jitter* yang terjadi lebih sedikit karena tidak ada atau lemahnya derau akibat interferensi jala-jala.

1.3 Kode

Berikut adalah kode *Octave* yang digunakan pada topik 1. Secara garis besar, kode ini membaca kedua sinyal sampel ECG dari file *.dat, kemudian menghitung FFT dan PSD dari masing-masing sinyal, dan menampilkan kedua sinyal dalam *domain* waktu dan PSD-nya dalam satuan dB. Kode ini tidak membutuhkan *package* eksternal dan menggunakan *built-in function* **fft**, **fftshift**, **abs**, dan **log10**.

Listing 1: Kode *Octave* untuk Topik 1: Analisis Sinyal

```
1  %{
2      Tugas Besar EB3102 — Pengolahan Sinyal Biomedika
3      Irfan Tito Kurniawan
4      NIM 18317019
5
6      Topik 1 — Analisis Sinyal
7
8      Built—in functions used:
9          — fft
10         — fftshift
11         — abs
12         — log10
13 %}
```

```
14
15 % Clear the screen, clean all the variables, close all windows
16 clear; clc; close all;
17
18 % Open the file
19 filename_1 = "psb_ecg1.dat";
20 filename_2 = "psb_ecg2.dat";
21
22 x_1 = load(filename_1);
23 x_2 = load(filename_2);
24
25 % Signal properties
26 sample_frequency = 1000;
27 power_supply_frequency = 60;
28 max_frequency = sample_frequency / 2;
29
30 signal_length_1 = length(x_1);
31 n_1 = [0 : signal_length_1 - 1];
32
33 signal_length_2 = length(x_2);
34 n_2 = [0 : signal_length_2 - 1];
35
36 % Generate the FFT and the PSD (in dB) of the signal
37 x_1_fft = fftshift(fft(x_1));
38 x_1_psd = 20 * log10(abs(x_1_fft) .^ 2);
39
40 x_2_fft = fftshift(fft(x_2));
41 x_2_psd = 20 * log10(abs(x_2_fft) .^ 2);
42
43 % Plot Stuffs
44 % ECG Sample 1
45 figure;
46 subplot(2, 1, 1);
47 plot(n_1, x_1);
48 title('ECG Sample 1 in Time Domain');
49 xlabel('n (sample)')
50 ylabel('x(n)')
51 xlim([0, signal_length_1])
52
53 subplot(2, 1, 2);
54 stem((n_1 / signal_length_1 * sample_frequency) - (sample_frequency / 2), x_1_psd);
55 title('ECG Sample 1 PSD');
56 xlabel('f (Hz)');
57 ylabel('Power (dB)')
58
59 % ECG Sample 2
60 figure;
```

```
61 subplot(2, 1, 1);
62 plot(n_2, x_2);
63 title('ECG Sample 1 in Time Domain');
64 xlabel('n (sample)')
65 ylabel('x(n)')
66 xlim([0, signal_length_2])
67
68 subplot(2, 1, 2);
69 stem((n_2 / signal_length_2 * sample_frequency) - (sample_frequency / 2), x_2_psd);
70 title('ECG Sample 1 PSD');
71 xlabel('f (Hz)');
72 ylabel('Power (dB)')
```

2 Topik II: Sistem Sebagai Filter

2.1 Teori

2.1.1 Notch dan Comb Filter

Notch Filter adalah sebuah filter yang memiliki satu atau lebih jurang, selanjutnya disebut dengan *notch*, pada respons magnitudanya [OH18]. Idealnya, *notch filter* menghilangkan komponen frekuensi tertentu dari sinyal tanpa memengaruhi komponen frekuensi lain dari sinyal. *Notch filter* berguna untuk keperluan yang membutuhkan kita untuk menghilangkan komponen frekuensi tertentu dari sinyal, seperti menghilangkan interferensi jala-jala beserta harmonisanya.

Untuk membuat *notch* pada respons magnituda pada frekuensi ω_0, kita hanya perlu menaruh sepasang *zero* yang bersifat konjugat kompleks dengan satu sama lain pada lingkaran satuan pada sudut ω_0 [OH18]. Yakni pada

$$z_{1,2} = e^{\pm j\omega_0} \tag{2}$$

Dengan meletakkan *pole* pada titik *origin*, kita memperoleh *notch filter* dengan *finite impulse response* (FIR) yang memiliki fungsi transfer berikut

$$H(z) = b_0(1 - e^{j\omega_0}z^{-1})(1 - e^{-j\omega_0}z^{-1}) = b_0(1 - 2\cos\omega_0 z^{-1} + z^{-2}) \tag{3}$$

Meskipun *notch filter* FIR memiliki respons fasa yang linear, *notch filter* FIR memiliki *bandwidth* yang cukup besar. Akibatnya, komponen frekuensi di sekitar *notch* ikut teredam. Pada respons frekuensinya, tampak bahwa *notch* yang terbentuk berbentuk landai.

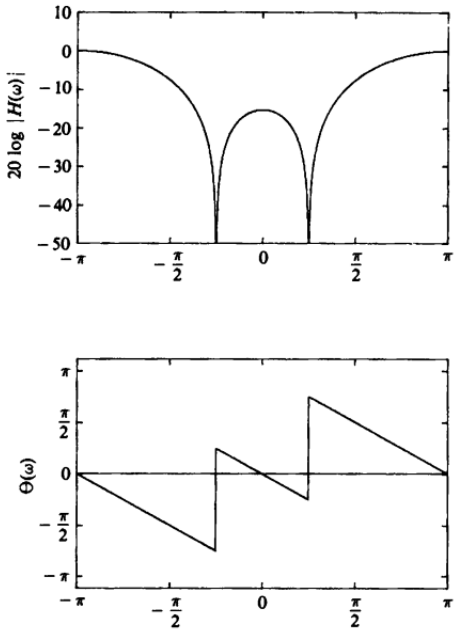

Figure 7: Respons frekuensi *notch filter* FIR dengan *notch* pada frekuensi $\omega = \frac{\pi}{4}$

Untuk mengurangi *bandwidth* dari *notch filter*, kita dapat meletakkan sepasang *pole* yang bersifat konjugat kompleks terhadap satu sama lain di dalam lingkaran satuan dengan sudut yang sama dengan *zero* yang telah diletakkan sebelumnya. Yakni pada

$$z_{1,2} = re^{\pm j\omega_0} \tag{4}$$

Dengan r merupakan bilangan real dengan nilai di antara 0 hingga 1 yang menunjukkan jarak *pole* dari titik *origin*. Dengan menambahkan *pole*, *bandwidth* dari *notch filter* berkurang, akan tetapi penambahan *pole* juga dapat menyebabkan sedikit *ripple* pada *passband* dari filter. Selain itu, respons fasa filter menjadi tidak linear dan filter memiliki *infinite impulse response* (IIR). Setelah penambahan *pole*, fungsi transfer dari *notch filter* menjadi

$$H(z) = b_0 \frac{1 - 2\cos\omega_0 z^{-1} + z^{-2}}{1 - 2r\cos\omega_0 z^{-1} + r^2 z^{-2}} \tag{5}$$

Sebagai contoh, berikut adalah plot respons frekuensi dari *notch filter* dengan *notch* pada $\frac{\pi}{4}$ setelah ditambahkan *pole* pada $r = 0.85$ dan $r = 0.95$

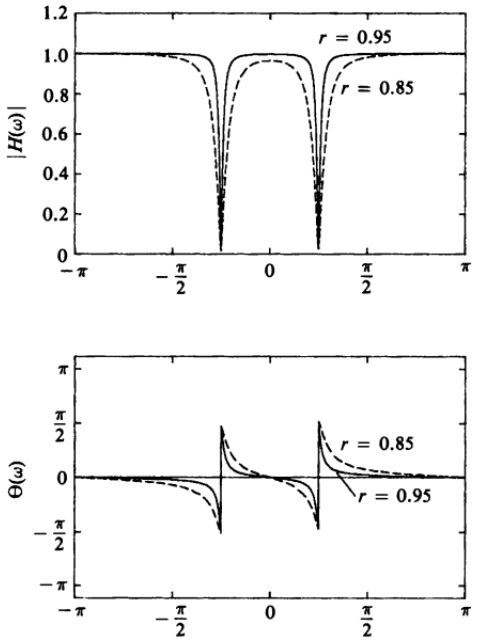

Figure 8: Respons frekuensi *notch filter* dengan *notch* pada frekuensi $\omega = \frac{\pi}{4}$ dan *pole* pada $r = 0.95$ dan $r = 0.85$

Untuk menghilangkan seluruh harmonisa dari derau akibat interferensi jala-jala, dapat digunakan **Comb Filter**. *Comb filter* dapat dibuat dengan merangkai beberapa *notch filter* pada frekuensi yang berkelipatan secara kaskade. Dengan demikian, diperoleh bentuk respons frekuensi yang menyerupai sebuah sisir, maka dari itu filter ini disebut *comb filter*.

2.1.2 Bandpass Filter

Bandpass Filter adalah jenis filter selektif frekuensi yang meloloskan sinyal dengan rentang frekuensi tertentu dan meredam sinyal di luar rentang tersebut. Berikut adalah respons magnituda dari *bandpass filter* ideal

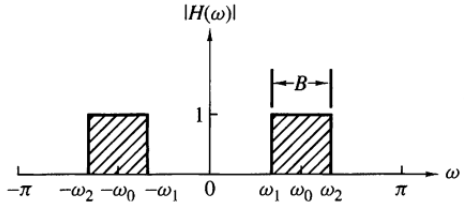

Figure 9: Respons frekuensi *bandpass filter* ideal

Bandpass filter dapat diimplementasikan dengan beberapa cara. Salah satunya adalah dengan merangkai *lowpass filter* yang memiliki frekuensi *cutoff* tinggi dengan *highpass filter* yang memiliki

frekuensi *cutoff* rendah secara kaskade. Dengan demikian, akan diperoleh respons frekuensi gabungan dari kedua filter tersebut, yang menghasilkan respons frekuensi seperti pada Gambar 9.

Terdapat beberapa pendekatan standar untuk merancang *bandpass filter* digital. Beberapa di antaranya adalah dengan merancang *bandpass filter* analog terlebih dahulu kemudian melakukan transformasi filter yang telah dirancang ke *domain* digital, dan dengan melakukan teknik *windowing*. Pada laporan ini, hanya akan dibahas pendekatan perancangan *bandpass filter* dengan merancang filter analog terlebih dahulu kemudian melakukan transformasi ke *domain* digital.

Pendekatan perancangan *bandpass filter* digital dengan merancang filter analog terlebih dahulu dilakukan karena ilmu perancangan filter analog yang lebih *well-established* dibandingkan dengan ilmu perancangan filter digital. Salah satu *template* filter yang umum adalah filter Butterworth dan Chebyshev. Filter Butterworth memiliki karakteristik respons frekuensi tanpa *ripple* pada *passband* dan *stopband*-nya dengan *trade-off transition band* yang landai dan panjang. Di sisi lain, filter Chebyshev memiliki *transition band* yang pendek namun dengan *ripple* pada *passband* dan *stopband*-nya.

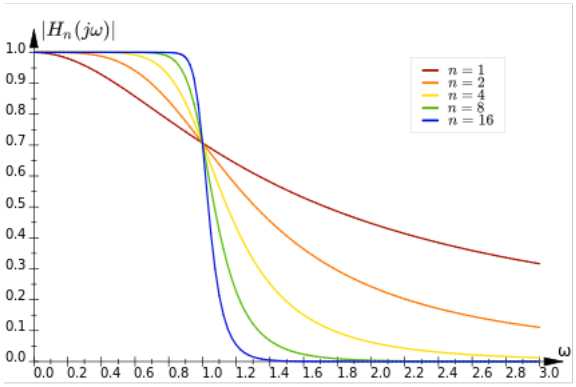

Figure 10: Respons frekuensi filter *lowpass* Butterworth ternormalisasi

Perancangan filter Butterworth pada umumnya dimulai dengan merancang *lowpass filter* yang ternormalisasi, yakni memiliki frekuensi *cutoff* pada $\omega_c = 1$ Hz kemudian dilakukan transformasi pada filter tersebut sehingga diperoleh filter dengan jenis dan frekuensi *cutoff* yang diinginkan.

Lowpass filter analog Butterworth berorde n dan frekuensi *cutoff* ω_c didefinisikan memiliki fungsi transfer berikut

$$|H_n(s)| = \frac{1}{\sqrt{1 + (\frac{s}{\omega_c})^{2n}}} = \frac{1}{B_n(\frac{s}{\omega_c})} \tag{6}$$

Sedangkan *highpass filter* analog Butterworth berorde n dan frekuensi *cutoff* ω_c didefinisikan memiliki fungsi transfer berikut

$$|H_n(s)| = \frac{1}{\sqrt{1 + (\frac{s}{\omega_c})^{-2n}}} = \frac{s^n}{\omega_c^n B_n(\frac{s}{\omega_c})} \tag{7}$$

Dengan B_n merupakan polinom Butterworth yang didefinisikan sebagai

$$B_n(s) = \begin{cases} \prod_{k=0}^{\frac{n}{2}-1}(s^2 - 2\cos 2\pi\frac{2k+n+1}{4n}s + 1) & n \text{ genap} \\ (s+1)\prod_{k=0}^{\frac{n-1}{2}-1}(s^2 - 2\cos 2\pi\frac{2k+n+1}{4n}s + 1) & n \text{ ganjil} \end{cases} \tag{8}$$

Setelah merancang *lowpass filter* dan *highpass filter* dengan frekuensi *cutoff* yang sesuai, fungsi transfer dari kedua filter dikalikan untuk memperoleh fungsi transfer *bandpass filter* Butterworth analog. Setelah itu, dapat dilakukan pendekatan digital atas *bandpass filter* analog yang telah dirancang, salah satunya adalah dengan transformasi Bilinear juga dikenal sebagai transformasi Tustin, yakni dengan mensubstitusikan

$$s \leftarrow \frac{2(z-1)}{T(z+1)} \tag{9}$$

Dengan T merupakan *sample time* dari sistem. Setelah dilakukan transformasi Tustin, diperoleh fungsi transfer filter *lowpass* Butterworth sebagai berikut

$$H_n(z) = \frac{Z_n(z)}{P_n(z)} \tag{10}$$

Dengan $Z_n(z)$ dan $P_n(z)$ masing-masing merupakan polinom *zero* dan *pole* dari filter yang didefinisikan sebagai

$$Z_n(z) = \begin{cases} \prod_{k=0}^{\frac{n}{2}-1}(1 + 2z^{-1} + z^{-2}) & n \text{ genap} \\ (1 - z^{-1})\prod_{k=0}^{\frac{n-1}{2}-1}(1 + 2z^{-1} + z^{-2}) & n \text{ ganjil} \end{cases} \tag{11}$$

$$P_n(z) = \begin{cases} \prod_{k=0}^{\frac{n}{2}-1}((\gamma^2 - \alpha\gamma + 1) + (-2\gamma^2 + 2)z^{-1} + (\gamma^2 + \alpha\gamma + 1)z^{-2}) & n \text{ genap} \\ ((\gamma+1) + (1-\gamma)z^{-1})\prod_{k=0}^{\frac{n-1}{2}-1}((\gamma^2 - \alpha\gamma + 1) + (-2\gamma^2 + 2)z^{-1} + (\gamma^2 + \alpha\gamma + 1)z^{-2}) & n \text{ ganjil} \end{cases} \tag{12}$$

Dengan γ dan α suatu konstanta yang didefinisikan sebagai

$$\gamma = \cot\frac{\omega_c\pi}{F_s} \tag{13}$$

$$\alpha = 2\cos 2\pi\frac{2k+n+1}{4n} \tag{14}$$

Dengan F_s merupakan *sample frequency* dari sistem. Sedangkan untuk filter *highpass* Butterworth, $Z_n(z)$ didefinisikan sebagai

$$Z_n(z) = \prod_{k=1}^{n}(\gamma - \gamma z^{-1}) \tag{15}$$

2.2 Perancangan Filter

2.2.1 Pengolahan Sinyal

Berdasarkan pembahasan pada topik 1, diperoleh bahwa pada kedua sinyal sampel ECG yang diberikan terdapat tiga jenis derau: derau berfrekuensi rendah, derau akibat interferensi jala-jala, dan derau berfrekuensi tinggi. Karena itu, diperlukan filter untuk menghilangkan ketiga jenis derau tersebut. Untuk derau berfrekuensi rendah dan tinggi akan dirancang sebuah *bandpass filter* dan untuk derau akibat interferensi jala-jala akan dirancang sebuah *comb filter*.

2.2.2 Perancangan *Bandpass Filter*

Sinyal ECG memiliki rentang frekuensi antara 0.5 Hz hingga 20 Hz. Karenanya, *bandpass filter* yang dirancang memiliki frekuensi *cutoff* pada $\omega_1 = 0.5$ Hz dan $\omega_2 = 20$ Hz. Dengan mengatur frekuensi *cutoff* pada nilai tersebut, *bandpass filter* akan mampu menghilangkan derau berfrekuensi DC dan rendah, derau berfrekuensi tinggi, serta membantu meredam derau akibat interferensi jala-jala.

Sinyal ECG memiliki informasi dalam *domain* waktu. Karena itu, penting bagi filter untuk tidak memiliki *ripple* pada respons frekuensinya agar perubahan bentuk sinyal dalam *domain* waktu dapat diminimalkan. Untuk itu, dipilihlah filter Butterworth karena respons frekuensinya yang *maximally flat* atau dengan kata lain tidak memiliki *ripple*.

Setelah memilih orde filter 4 untuk *lowpass* dan *highpass filter*, dengan frekuensi *cutoff* di atas diperoleh *pole-zero plot* dari *bandpass filter* sebagai berikut

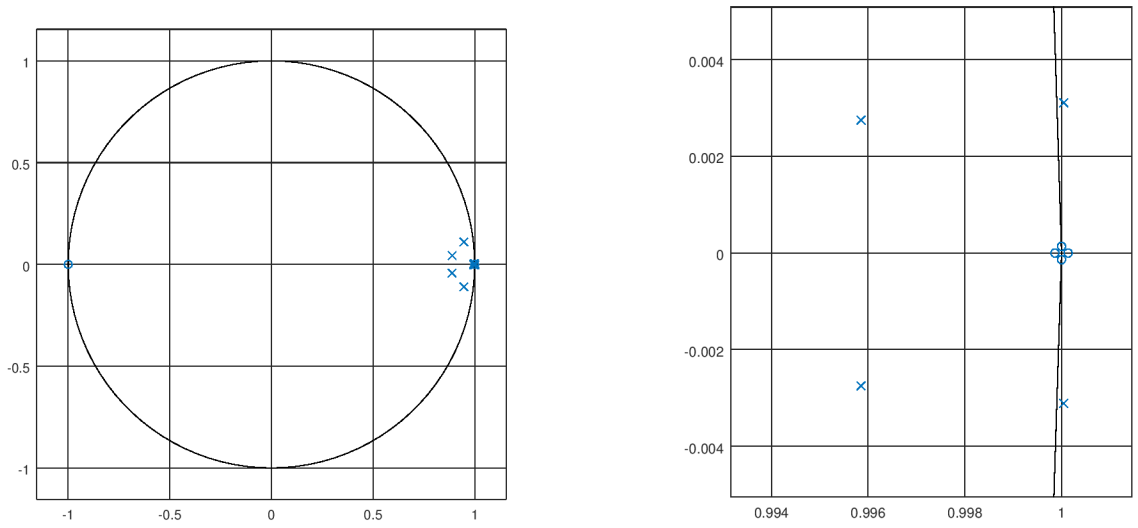

Figure 11: *Pole-zero plot* dari *bandpass filter* Butterworth orde 4 dengan $\omega_1 = 0.5$ Hz dan $\omega_2 = 20$ Hz

Filter yang diperoleh memiliki delapan buah *pole* di sekitar $\omega = 0$ dan delapan buah *zero* dengan empat buah *zero* di sekitar $\omega = 0$ dan empat buah *zero* di $\omega = \pi$.

Filter ini memiliki respons frekuensi sebagai berikut:

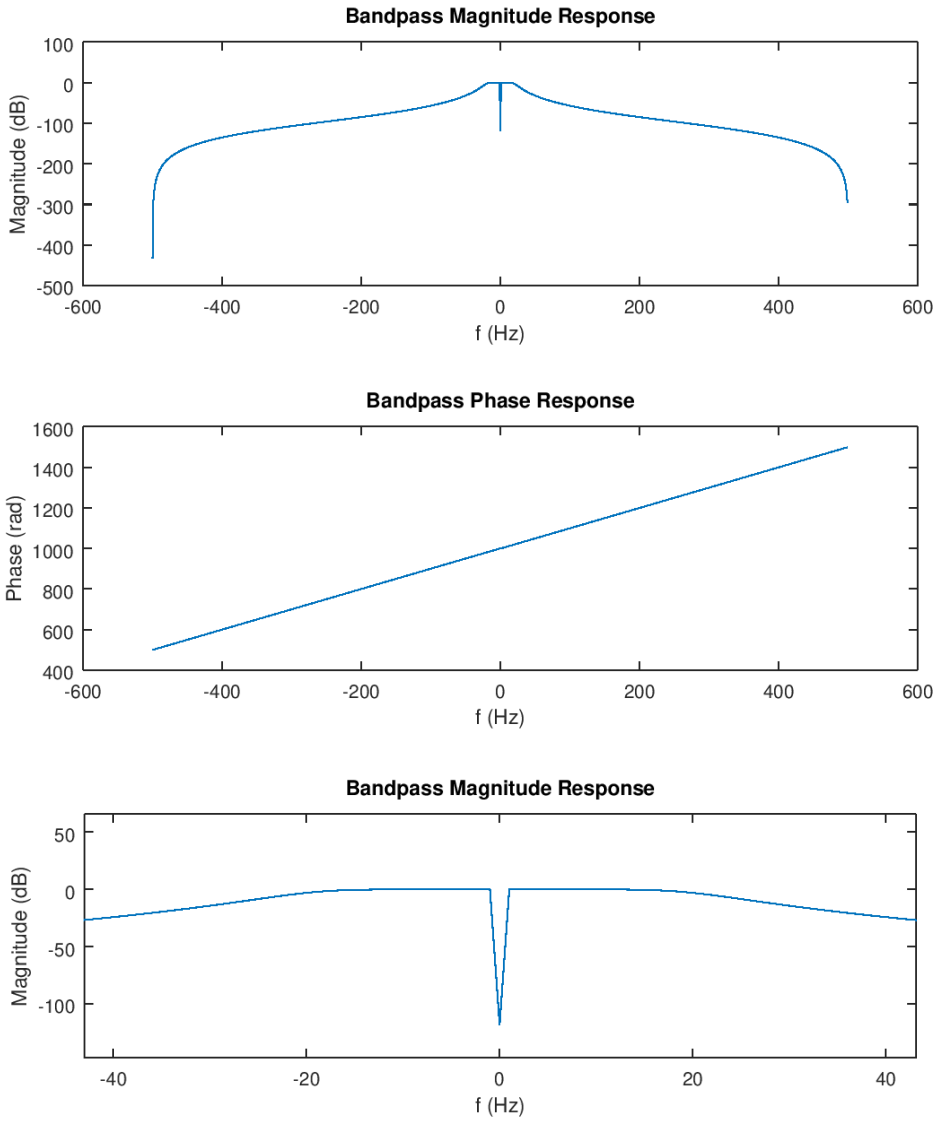

Figure 12: Respons frekuensi *bandpass filter* Butterworth orde 4 dengan $\omega_1 = 0.5$ Hz dan $\omega_2 = 20$ Hz

Dari plot respons frekuensi *bandpass filter*, terlihat bahwa *bandpass filter* berperilaku seperti seharusnya dan memang memiliki frekuensi *cutoff* pada $\omega_1 = 0.5$ Hz dan $\omega_2 = 20$ Hz. *bandpass filter* melemahkan dengan tajam komponen sinyal berfrekuensi di bawah 0.5 Hz dan melemahkan secara relatif tidak terlalu tajam komponen sinyal berfrekuensi di atas 20 Hz.

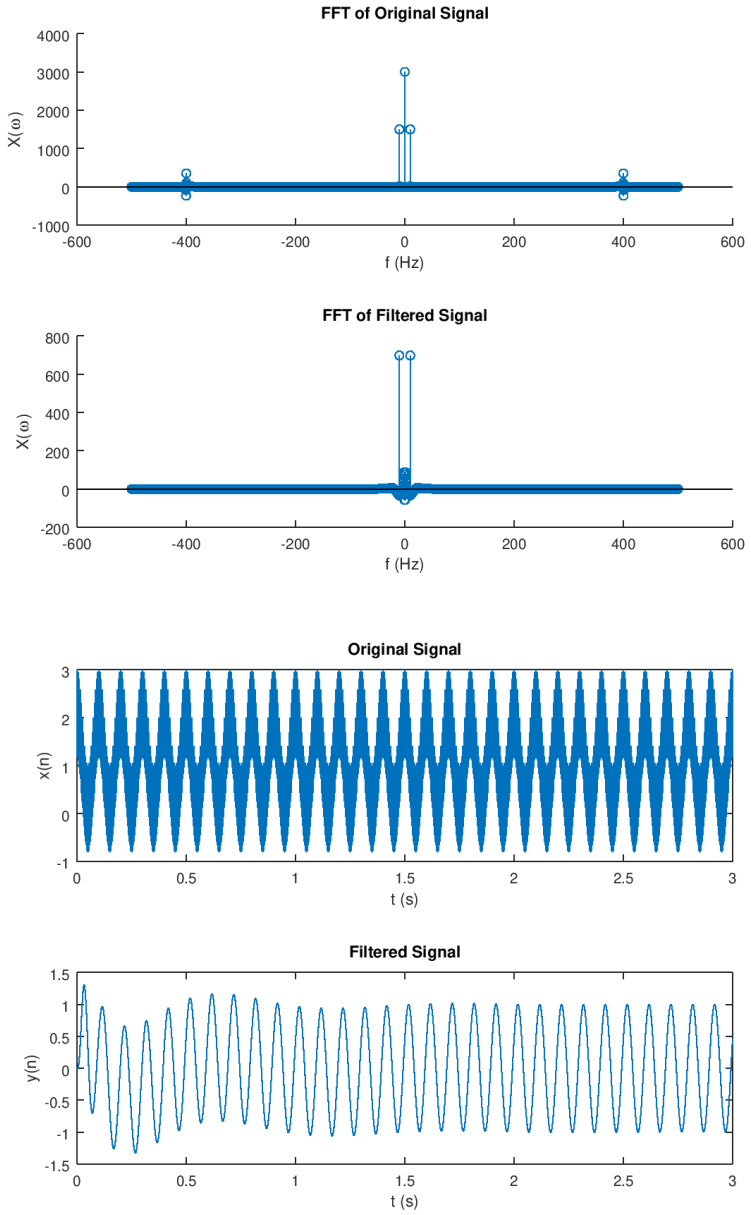

Figure 13: Hasil uji coba filter *bandpass* pada sinyal cosinus gabungan frekuensi 0.01 Hz, 10 Hz, dan 400 Hz dan FFT-nya

Untuk menguji coba filter ini, digunakan sebuah sinyal input yang terdiri atas tiga buah sinyal cosinus yang digabungkan, masing-masing berfrekuensi 0.01 Hz, 10 Hz, dan 400 Hz. Seperti yang terlihat pada hasil FFT dari sinyal, filter berhasil meredam komponen cosinus dengan frekuensi 0.01 Hz dan 400 Hz, menyisakan komponen cosinus yang memiliki frekuensi pada *passband* dari filter, yakni pada 10 Hz.

Pada awal sinyal hasil filter pada *domain* waktu, tampak sedikit fluktuasi *baseline* dari sinyal. Hal ini dikarenakan proses *filtering* yang dilakukan mengasumsikan kondisi rileks, sehingga butuh waktu agar komponen cosinus berfrekuensi rendah hilang. Selain itu, juga tampak bahwa *jitter* pada sinyal, yang disebabkan oleh komponen cosinus berfrekuensi tinggi, hilang pada sinyal hasil *filtering*. Setelah beberapa saat, sinyal hasil *filtering* berbentuk cosinus dengan frekuensi 10 Hz murni, seperti yang diharapkan.

2.2.3 Perancangan *Comb Filter*

Berdasarkan hasil analisis sinyal pada topik 1, diperoleh bahwa sinyal sampel ECG yang diberikan memiliki derau akibat interferensi jala-jala pada frekuensi 60 Hz beserta harmonisanya. Untuk menghilangkan derau tersebut, dirancang sebuah *notch filter* untuk setiap frekuensi harmonisa dari frekuensi jala-jala. Karena itu, diperlukan notch filter sejumlah

$$n = \frac{F_s}{f_{jala-jala}} = \frac{1000}{60} = 16 \tag{16}$$

Sehingga akan diperoleh *comb filter* dengan 32 buah *notch*, 32 buah *pole* dan 32 buah *zero* pada sudut kelipatan

$$\omega = \frac{f_{jala-jala}}{F_s} = \frac{60}{1000} = 0.06 \text{ /siklus} = 0.3769 \text{ rad/s} \tag{17}$$

Dengan mengambil jarak *pole* dari origin sebesar $r = 0.995$, diperoleh *pole-zero plot* berikut

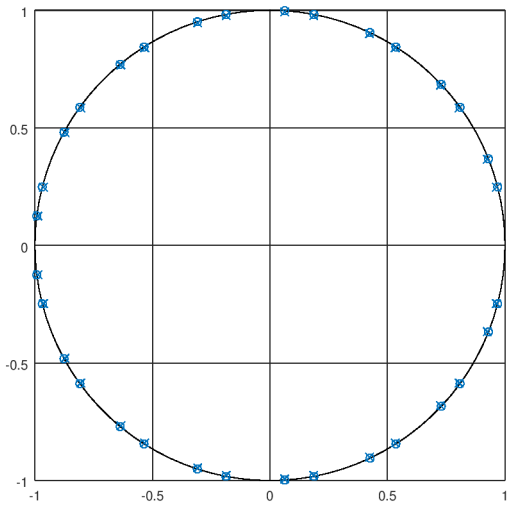

Figure 14: *Pole-zero plot* dari *comb filter*

Filter ini memiliki respons frekuensi sebagai berikut

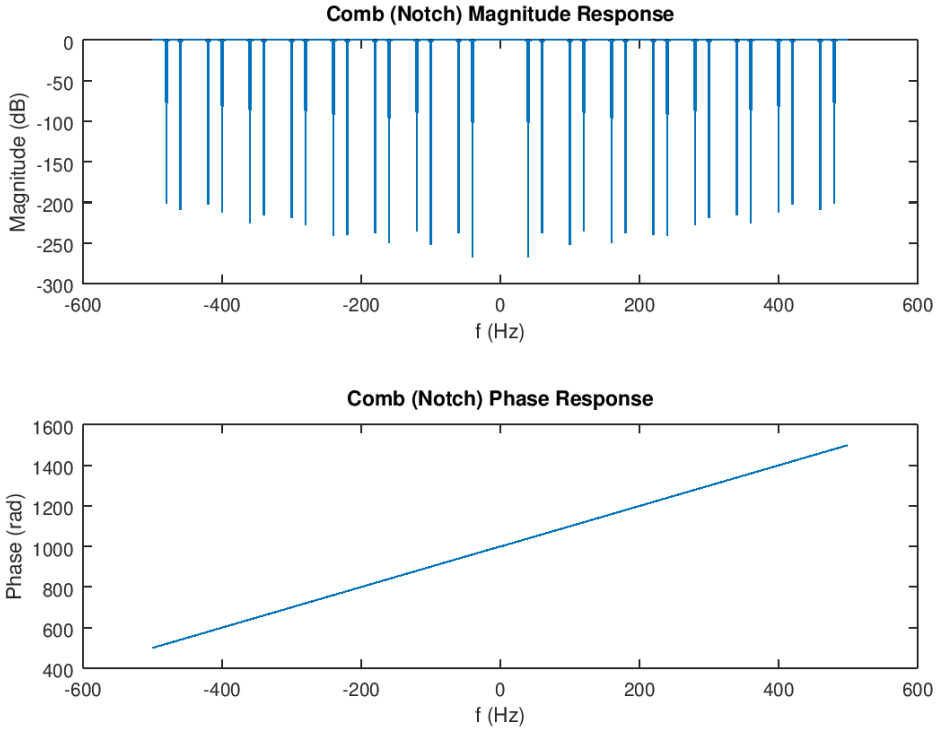

Figure 15: Respons frekuensi dari *comb filter*

Pada respons magnituda dari filter, terdapat 32 buah *notch* pada frekuensi 60 Hz beserta dengan kelipatannya. *Notch* pada frekuensi lebih dari setengah *sample frequency*, 500 Hz, mengalami *frequency folding* ke frekuensi di bawah 500 Hz.

Meskipun merupakan filter IIR, respons fasa dari *comb filter* yang diperoleh bersifat linear. Sifat ini penting karena sinyal ECG memiliki informasi dalam *domain* waktu. Respons fasa yang linear menggeser tiap komponen frekuensi dari sinyal ECG sejauh suatu besaran waktu yang sama besar di *domain* waktu sehingga bentuk dari sinyal akan terjaga.

Untuk menguji filter yang dirancang, diberikan sinyal *input* yang terdiri atas penjumlahan tiga buah sinyal cosinus berfrekuensi masing-masing 10 Hz, 60 Hz, dan 120 Hz. Pemilihan frekuensi komponen sinyal cosinus ini dikarenakan frekuensi 10 Hz termasuk dalam rentang frekuensi *passband* dari filter, sedangkan frekuensi 60 Hz dan 120 Hz merupakan frekuensi salah satu *notch* dari filter.

Berikut adalah hasil pengujian *comb filter* dengan sinyal *input* tersebut

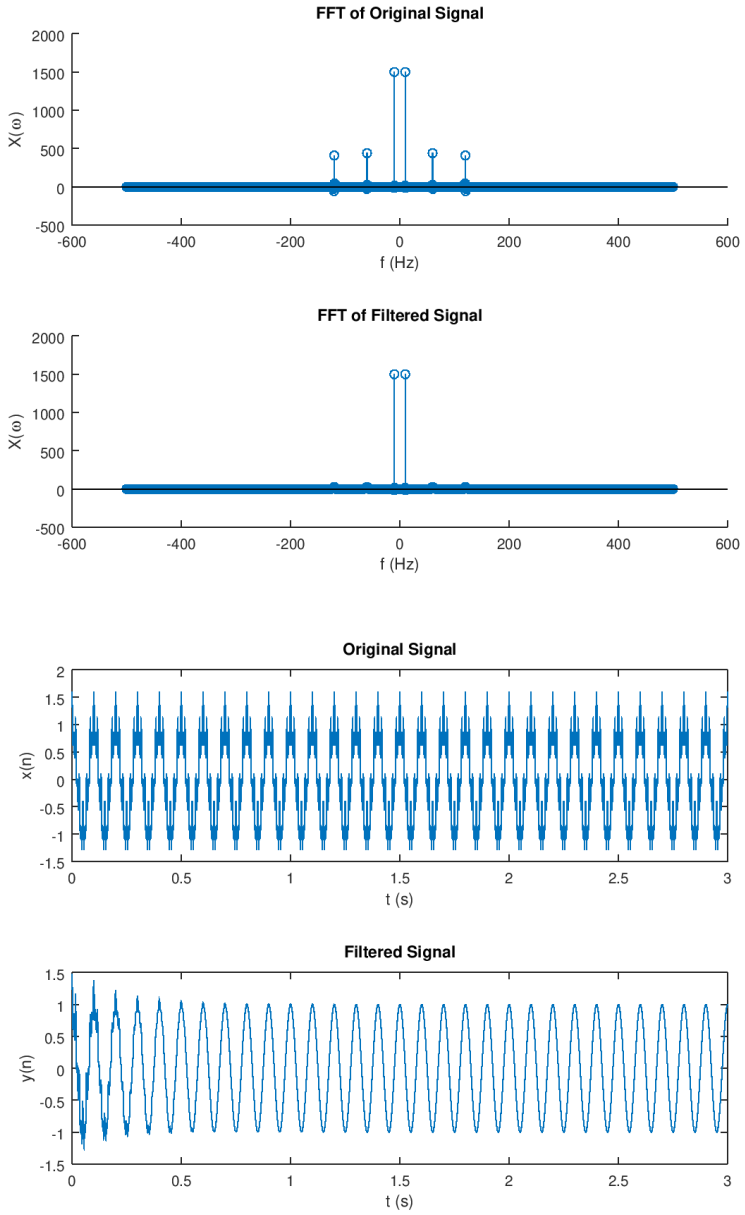

Figure 16: Hasil uji coba *comb filter* pada sinyal cosinus gabungan frekuensi 10 Hz, 60 Hz, dan 120 Hz dan FFT-nya

Hasil FFT dari sinyal *input* dan sinyal yang telah difilter menunjukkan bahwa *notch* pada frekuensi 60 Hz dan 120 Hz bekerja dengan efektif untuk menghilangkan komponen frekuensi tersebut dari sinyal gabungan. Kemudian, terlihat bahwa komponen frekuensi 10 Hz dari sinyal gabungan tidak mengalami perubahan magnituda. Hal ini menunjukkan bahwa *comb filter* tidak meredam komponen frekuensi pada rentang frekuensi *passband*-nya.

Pada sinyal *domain* waktu, tampak bahwa seiring berjalannya waktu, sinyal berbentuk *cosinus* murni berfrekuensi 10 Hz. Hal ini menunjukkan bahwa *comb filter* bekerja seperti seharusnya. Namun, pada awalnya, sinyal hasil *filtering* terlihat mengandung derau. Sama seperti pada pengujian *bandpass filter*, hal ini dikarenakan proses *filtering* mengasumsikan kondisi awal rileks, sedangkan kondisi awal sebenarnya tidak demikian.

2.3 Hasil dan Pembahasan

Diberikan dua sinyal sampel ECG yang dianalisis pada topik 1. Kedua sinyal sampel ECG tersebut akan difilter dengan kedua filter yang telah dirancang pada bagian sebelumnya yang dirangkai secara kaskade. Berikut adalah PSD sinyal mentah dan sinyal hasil proses *filtering* yang dilakukan pada sinyal sampel ECG 1

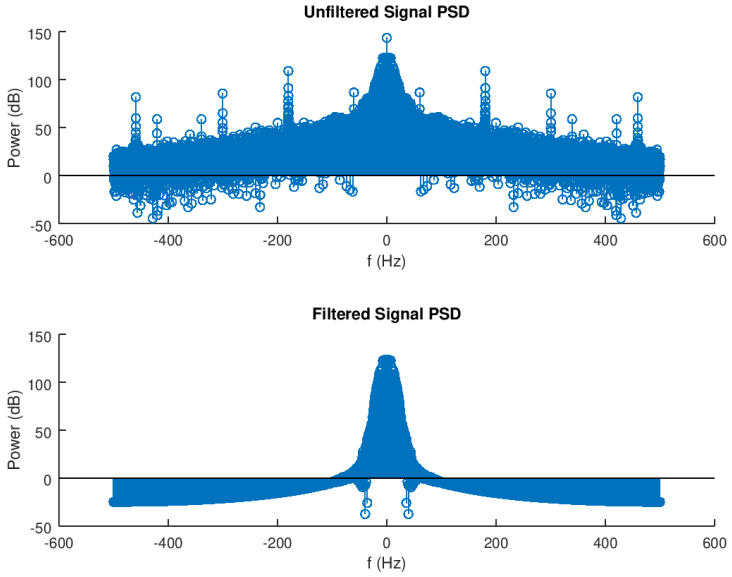

Figure 17: PSD hasil proses *filtering* terhadap sinyal sampel ECG 1

Sebelum dilakukan proses *filtering*, PSD sinyal memiliki banyak lonjakan daya pada frekuensi jala-jala dan kelipatannya serta pada frekuensi rendah. Selain itu, daya dari sinyal tersebar pada seluruh rentang frekuensi yang ada.

Setelah mengalami proses *filtering*, lonjakan daya pada frekuensi jala-jala dan kelipatannya hilang. Hal ini dikarenakan penggunaan *comb filter* yang meredam komponen frekuensi jala-jala dan kelipatannya dari sinyal dan sesedikit mungkin memengaruhi daerah di sekitarnya. Kemudian, peredaman komponen frekuensi jala-jala dan kelipatannya turut dibantu oleh penggunaan *bandpass filter* yang memiliki frekuensi *cutoff lowpass* pada frekuensi 20 Hz, lebih rendah dibandingkan harmonisa pertama dari frekuensi jala-jala yakni 60 Hz sehingga seluruh harmonisa dari frekuensi jala-jala

turut teredam. Hasil dari peredaman ini tampak pada PSD sinyal yang telah melewati proses *filtering* yang halus dan tidak memiliki lonjakan daya pada frekuensi tertentu.

Selain itu, setelah mengalami proses *filtering*, lonjakan daya pada frekuensi rendah dan frekuensi DC dan sebaran daya pada frekuensi tinggi turut hilang. Hal ini dikarenakan penggunaan *bandpass filter* yang memblok komponen sinyal berfrekuensi di bawah 0.5 Hz dan di atas 20 Hz. Akibatnya, derau frekuensi rendah hilang dan derau pada frekuensi tinggi diredam sehingga diperoleh PSD sinyal yang terkonsentrasi pada rentang frekuensi sinyal ECG, yakni dari 0.5 Hz hingga 20 Hz seperti yang diharapkan.

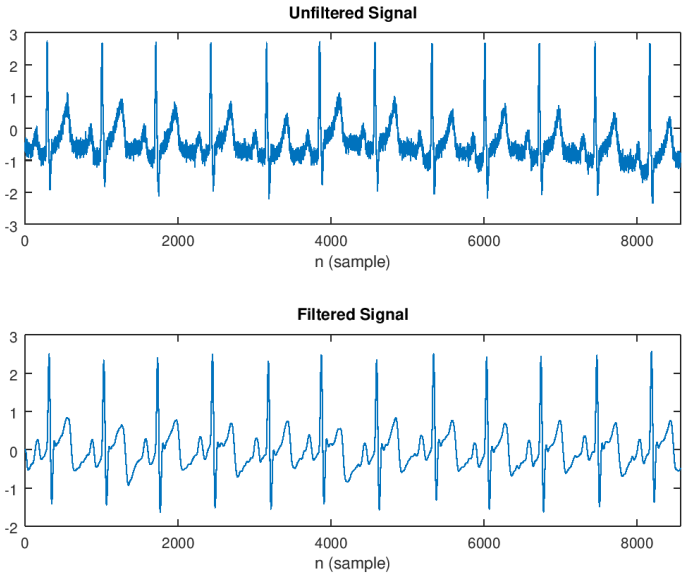

Figure 18: Hasil proses *filtering* terhadap sinyal sampel ECG 1 dalam *domain* waktu

Sinyal dalam *domain* waktu sebelum proses *filtering* tampak memiliki sedikit pergeseran *baseline* dan mengalami banyak *jitter*. Setelah mengalami proses *filtering*, diperoleh sinyal dengan *baseline* yang relatif lebih sedikit. Selain itu, sinyal hasil proses *filtering* mengalami *jitter* yang jauh lebih sedikit atau bahkan tidak ada, dan diperoleh sinyal yang sangat halus sesuai dengan *template* sinyal ECG.

Kemudian, berikut adalah PSD sinyal mentah dan sinyal hasil proses *filtering* yang dilakukan pada sinyal sampel ECG 2

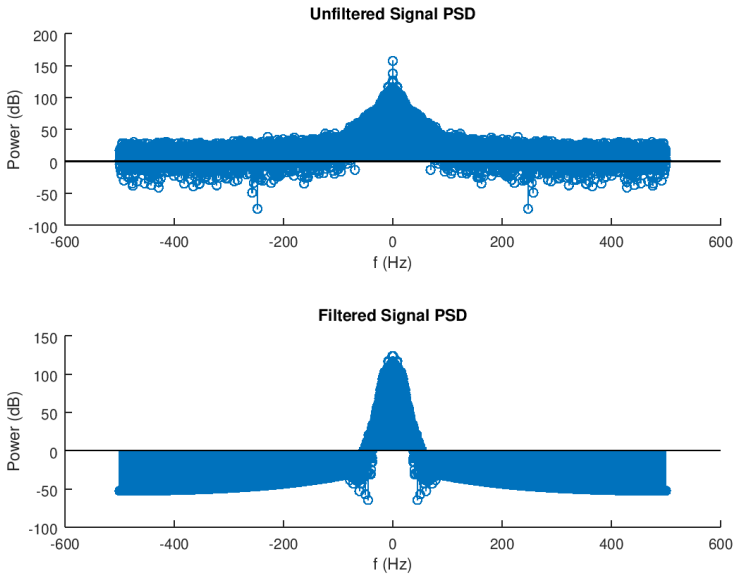

Figure 19: PSD hasil proses *filtering* terhadap sinyal sampel ECG 2

Berbeda dengan sampel sebelumnya, PSD sampel ECG 2 tidak memiliki lonjakan daya yang terlihat pada frekuensi jala-jala dan harmonisanya, namun PSD sampel ECG 2 memiliki derau berfrekuensi rendah yang jauh lebih kuat dibandingkan sampel sebelumnya. Seperti pada sampel sebelumnya, setelah mengalami proses *filtering*, berkat *bandpass filter*, tampak bahwa derau berfrekuensi rendah mengalami peredaman dan memiliki daya yang wajar seperti pada frekuensi di sekitarnya. Daya yang sebelumnya tersebar pada seluruh rentang frekuensi pun kini menjadi terkonsentrasi pada rentang frekuensi sinyal ECG saja, yakni pada 0.5 Hz hingga 20 Hz seperti yang diharapkan.

Dalam *domain* waktu, perbedaan antara sinyal sampel ECG 2 mentah dengan sinyal hasil proses *filtering* tampak jelas. Pada sinyal hasil proses *filtering*, tidak terdapat pergeseran *baseline* yang berarti dibandingkan dengan sinyal sampel mentah yang mengalami pergeseran *baseline* yang ekstrem. Hal ini disebabkan komponen berfrekuensi rendah dari sinyal mentah telah diredam oleh *bandpass filter*. Kemudian, *jitter* yang muncul pada sinyal mentah pun tidak ditemukan pada sinyal hasil proses *filtering*. Hal ini dikarenakan komponen berfrekuensi tinggi dari sinyal telah diredam pula oleh *bandpass filter*, menyisakan komponen berfrekuensi dalam rentang frekuensi sinyal ECG, sehingga diperoleh bentuk sinyal ECG yang menyerupai *template*.

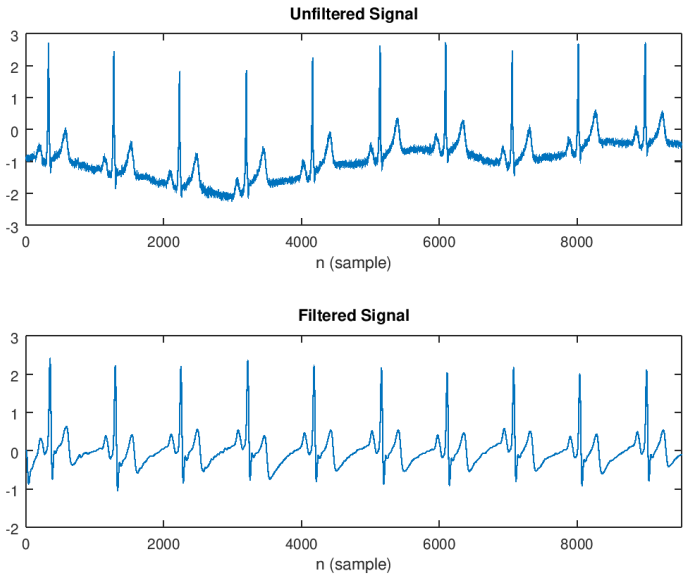

Figure 20: Hasil proses *filtering* terhadap sinyal sampel ECG 2 dalam *domain* waktu

2.4 Kode

Berikut adalah kode *Octave* yang digunakan pada topik 2. Ketika dijalankan, kode ini akan menampilkan **pole-zero plot** filter yang digunakan, **respons frekuensi** filter yang digunakan, **PSD** sinyal sebelum dan sesudah proses *filtering*, serta **sinyal dalam *domain* waktu** sebelum dan sesudah proses *filtering* dari kedua buah sinyal yang diberikan.

Kode ini menggunakan *package* signal untuk menampilkan *pole-zero plot* dari filter yang dirancang. Selain itu, kode ini tidak membutuhkan *package* eksternal lainnya. Perancangan persamaan perbedaan dari *comb filter* dan *bandpass filter* pada kode ini diimplementasikan dengan fungsi buatan sendiri bernama **design_comb** dan **design_butterworth_bandpass**. Selain itu, kode ini menggunakan *built-in function* **fft**, **fftshift**, **abs**, **unwrap**, **filter**, **freqz**, dan **conv**.

Listing 2: Kode *Octave* untuk Topik 2: Sistem Sebagai Filter

```
1  %{
2      Tugas Besar EB3102 — Pengolahan Sinyal Biomedika
3      Irfan Tito Kurniawan
4      NIM 18317019
5
6      Topik 2 — Sistem sebagai Filter
7
8      Built—in functions:
9          — fft
```

```
10          - fftshift
11          - abs
12          - unwrap
13          - filter
14          - freqz
15          - conv
16          - zplane (pkg signal)
17  %}
18
19  % Clear the screen, clean all the variables, close all windows
20  clear; clc; close all;
21
22  % Load signal for zplane function
23  pkg load signal
24
25  %{
26      Function definitions:
27          - design_notch: to design notch at a certain frequency
28          - design_comb: to design comb at a certain frequency multiples
29          - design_butterworth_lowpass: to design a nth order butterworth LPF
30          - design_butterworth_highpass: to design a nth order butterworth HPF
31          - design_butterworth_bandpass: to design a nth order butterworth BPF
32  %}
33
34  % Second order notch filter difference equation generator
35  function [b, a] = design_notch(notch_frequency, max_frequency, ...
        pole_distance_from_origin)
36      notch_frequency_ratio = notch_frequency / max_frequency;
37      cos_omega = cos(notch_frequency_ratio * pi);
38
39      % Difference equation
40      a_0 = 1;
41      a_1 = -2 * pole_distance_from_origin * cos_omega;
42      a_2 = pole_distance_from_origin .^ 2;
43      b_0 = 1;
44      b_1 = -2 * cos_omega;
45      b_2 = 1;
46
47      % Multiply with this so the gain is 1 at the passband
48      b = ((pole_distance_from_origin .^ 2 + (2 * pole_distance_from_origin) + 1) / ...
            4) .* [b_0, b_1, b_2];
49      a = [a_0, a_1, a_2];
50  end
51
52  function [b, a] = design_comb(natural_frequency, sample_frequency, ...
        pole_distance_from_origin)
53      b = [1];
```

```
54      a = [1];
55
56      % Calculate the needed notch amount
57      max_frequency = sample_frequency / 2;
58      frequency_ratio = natural_frequency / sample_frequency;
59      notch_amount = floor(1 / frequency_ratio);
60
61      % Convolve all the notch filter difference equation
62      for i = 1:notch_amount
63          [dummy_b, dummy_a] = design_notch(i * natural_frequency, max_frequency, ...
                pole_distance_from_origin);
64          b = conv(b, dummy_b);
65          a = conv(a, dummy_a);
66      end
67  end
68
69  % Butterworth lowpass filter difference equation generator
70  function [b, a] = design_butterworth_lowpass(n, cutoff_frequency, sample_frequency)
71      a = [1];
72      b = [1];
73
74      % Constant for discretization
75      gamma_const = cot(pi * cutoff_frequency / sample_frequency);
76
77      if mod(n, 2) == 0
78          for k = 0:(n / 2) - 1
79              % Constant for each pole
80              alpha_const = 2 * cos(2 * pi * (2 * k + n + 1) / (4 * n));
81              dummy_a = [gamma_const .^ 2 - (alpha_const * gamma_const) + 1, -2 * ...
                    (gamma_const .^ 2) + 2, ...
82                  gamma_const .^ 2 + (alpha_const * gamma_const) + 1];
83              a = conv(a, dummy_a);
84
85              dummy_b = [1, 2, 1];
86              b = conv(b, dummy_b);
87          end
88      else
89          for k = 0:((n - 1) / 2) - 1
90              alpha_const = 2 * cos(2 * pi * (2 * k + n + 1) / (4 * n));
91              dummy_a = [gamma_const .^ 2 - (alpha_const * gamma_const) + 1, -2 * ...
                    (gamma_const .^ 2) + 2, ...
92                  gamma_const .^ 2 + (alpha_const * gamma_const) + 1];
93              a = conv(a, dummy_a);
94
95              dummy_b = [1, 2, 1];
96              b = conv(b, dummy_b);
97          end
```

```
98
99          dummy_a = [gamma_const + 1, 1 - gamma_const];
100         a = conv(a, dummy_a);
101
102         dummy_b = [1, 1];
103         b = conv(b, dummy_b);
104     end
105 end
106
107 % Butterworth highpass filter difference equation generator
108 function [b, a] = design_butterworth_highpass(n, cutoff_frequency, sample_frequency)
109     % Use lowpass filter generator to get the poles
110     [b, a] = design_butterworth_lowpass(n, cutoff_frequency, sample_frequency);
111
112     gamma_const = cot(pi * cutoff_frequency / sample_frequency);
113
114     % Redefine the zeros
115     b = [1];
116
117     for k = 1:n
118         dummy_b = gamma_const .* [1, -1];
119         b = conv(b, dummy_b);
120     end
121 end
122
123 % Butterworth bandpass filter difference equation generator
124 function [b, a] = design_butterworth_bandpass(n_lower, n_upper, ...
        cutoff_frequency_lower, cutoff_frequency_upper, sample_frequency)
125     % Generate lowpass and highpass filter difference equation
126     [b_low, a_low] = design_butterworth_lowpass(n_upper, cutoff_frequency_upper, ...
            sample_frequency);
127     [b_high, a_high] = design_butterworth_highpass(n_lower, ...
            cutoff_frequency_lower, sample_frequency);
128
129     % Combine both filters into one
130     b = conv(b_low, b_high);
131     a = conv(a_low, a_high);
132 end
133
134 % Signal processing
135 % Open the file
136 filename_1 = "psb_ecg1.dat";
137 filename_2 = "psb_ecg2.dat";
138
139 x_1 = load(filename_1);
140 x_2 = load(filename_2);
141
```

```
% Signal properties
sample_frequency = 1000;
power_supply_frequency = 60;
max_frequency = sample_frequency / 2;

signal_length_1 = length(x_1);
n_1 = [0 : signal_length_1 - 1];

signal_length_2 = length(x_2);
n_2 = [0 : signal_length_2 - 1];

% Generate the FFT and the PSD (in dB) of the signal
x_1_fft = fftshift(fft(x_1));
x_1_psd = 20 * log10(abs(x_1_fft) .^ 2);

x_2_fft = fftshift(fft(x_2));
x_2_psd = 20 * log10(abs(x_2_fft) .^ 2);

% Comb Filter
pole_distance_from_origin = 0.995;
[b_comb, a_comb] = design_comb(power_supply_frequency, sample_frequency, ...
    pole_distance_from_origin);

figure;
zplane(b_comb, a_comb)

% Plot the comb filter's frequency response
[h_comb, w_comb] = freqz(b_comb, a_comb, sample_frequency, 'whole', ...
    sample_frequency);

h_comb = fftshift(h_comb);
w_comb = fftshift(w_comb);

figure;
subplot(2, 1, 1);
plot(-max_frequency:max_frequency - 1, 20 * log10(abs(h_comb)));
xlabel('f (Hz)');
ylabel('Magnitude (dB)');
title('Comb (Notch) Magnitude Response');
subplot(2, 1, 2);
plot(-max_frequency:max_frequency - 1, unwrap(w_comb));
xlabel('f (Hz)');
ylabel('Phase (rad)');
title('Comb (Notch) Phase Response');

% Bandpass Filter
high_pass_frequency = 0.5;
```

```
low_pass_frequency = 20;
bandpass_order = 4;

[b_bandpass, a_bandpass] = design_butterworth_bandpass(bandpass_order, ...
    bandpass_order, ...
    high_pass_frequency, low_pass_frequency, sample_frequency);

figure;
zplane(b_bandpass, a_bandpass)

% Plot the bandpass filter's frequency response
[h_bandpass, w_bandpass] = freqz(b_bandpass, a_bandpass, sample_frequency, ...
    'whole', sample_frequency);

h_bandpass = fftshift(h_bandpass);
w_bandpass = fftshift(w_bandpass);

figure;
subplot(2, 1, 1);
plot(-max_frequency:max_frequency - 1, 20 * log10(abs(h_bandpass)));
xlabel('f (Hz)');
ylabel('Magnitude (dB)');
title('Bandpass Magnitude Response');
subplot(2, 1, 2);
plot(-max_frequency:max_frequency - 1, unwrap(w_bandpass));
xlabel('f (Hz)');
ylabel('Phase (rad)');
title('Bandpass Phase Response');

% Filter out the power line harmonic noises
y_1 = filter(b_comb, a_comb, x_1);
y_2 = filter(b_comb, a_comb, x_2);

% Filter out high and DC frequencies
y_1 = filter(b_bandpass, a_bandpass, y_1);
y_2 = filter(b_bandpass, a_bandpass, y_2);

% Check the resulting signal's PSD
y_1_fft = fftshift(fft(y_1));
y_1_psd = 20 * log10(abs(y_1_fft) .^ 2);

y_2_fft = fftshift(fft(y_2));
y_2_psd = 20 * log10(abs(y_2_fft) .^ 2);

% Plot time-domain signals
% ECG Sample 1
figure;
```

```
subplot(2, 1, 1);
plot(n_1, x_1);
xlabel('n (sample)');
title('Unfiltered Signal');
xlim([0, signal_length_1]);

subplot(2, 1, 2);
plot(n_1, y_1);
xlabel('n (sample)');
title('Filtered Signal');
xlim([0, signal_length_1]);

figure;
subplot(2, 1, 1);
stem((n_1 / signal_length_1 * sample_frequency) - (sample_frequency / 2), x_1_psd);
xlabel('f (Hz)');
ylabel('Power (dB)')
title('Unfiltered Signal PSD');

subplot(2, 1, 2);
stem((n_1 / signal_length_1 * sample_frequency) - (sample_frequency / 2), y_1_psd);
xlabel('f (Hz)');
ylabel('Power (dB)')
title('Filtered Signal PSD');

% ECG Sample 2
figure;
subplot(2, 1, 1);
plot(n_2, x_2);
xlabel('n (sample)');
title('Unfiltered Signal');
xlim([0, signal_length_2]);

subplot(2, 1, 2);
plot(n_2, y_2);
xlabel('n (sample)');
title('Filtered Signal');
xlim([0, signal_length_2]);

figure;
subplot(2, 1, 1);
stem((n_2 / signal_length_2 * sample_frequency) - (sample_frequency / 2), x_2_psd);
xlabel('f (Hz)');
ylabel('Power (dB)')
title('Unfiltered Signal PSD');

subplot(2, 1, 2);
```

```
279 stem((n_2 / signal_length_2 * sample_frequency) - (sample_frequency / 2), y_2_psd);
280 xlabel('f (Hz)');
281 ylabel('Power (dB)')
282 title('Filtered Signal PSD');
```

References

[ASSK14] U Rajendra Acharya, Jasjit Suri, Jos Spaan, and S. Krishnan. *Advances in Cardiac Signal Processing*. 10 2014.

[GS12] Mack Grady and Surya Santoso. Understanding power system harmonics. 2012.

[OH18] Alan Oppenheim and Peter Hagelstein. *Signals, Systems and Inference*. Massachusetts Institute of Technology: MIT OpenCourseWare, 2018.

[TWT84] Nitish V Thakor, John G Webster, and Willis J Tompkins. *Estimation of QRS Complex Power Spectra for Design of a QRS Filter*, volume BME-31. 11 1984.