

DSP en beeldverwerking [DSPIM]

**Project 2: Notch filter**

        Auteurs:

Jasper Coekaerts

Luka Lageschaar

Docent:

Bart Vanrumste

Inleiding

Het doel van het project is het ontwerp, implementatie en gebruik van een Notch filter. Om dit doelt te bereiken werden 2 ECG signalen met interferentie gegeven, ecg.mat en ecg2.mat. Deze signalen tonen de elektrische activiteit van het hart met een frequentie van respectievelijk 1000 Hz en 204,73 Hz. De Notch filter heeft de vorm van:

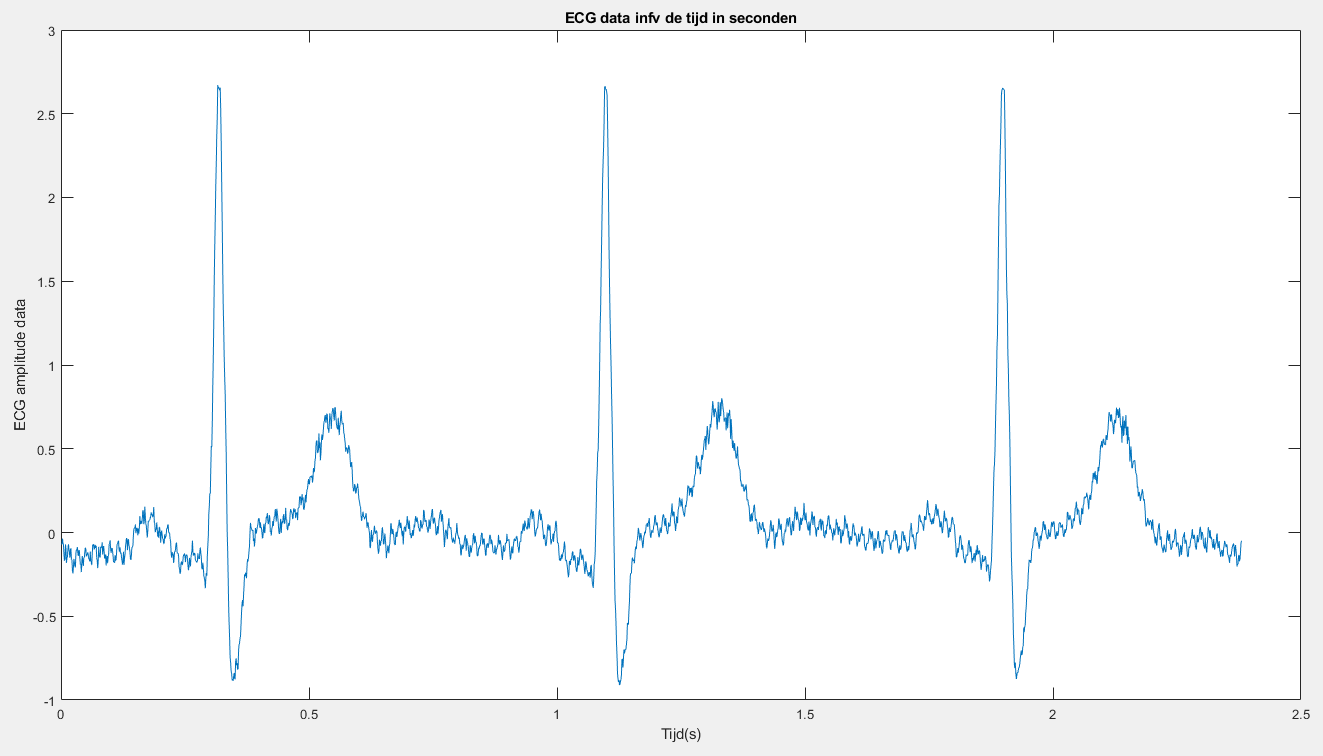
met a gelijk aan 0.9.

De signalen worden eerst gevisualiseerd door in Matlab de signalen uit te zetten in functie van de tijd in seconden. Daarna worden de signalen geanalyseerd om de spectrum van het signaal te bepalen en de power-line noise in dat spectrum te identificeren. De gegeven Notch filter wordt uitgeschreven in zijn differentievergelijking en daaruit de directe vorm II uitgehaald. Op papier worden de impuls en frequentie respons berekend voor 3 frequenties en de berekende waarden worden gecontroleerd met de resultaten berekend in Matlab. De filter wordt dan toegepast op de gegeven ECG signalen en de bijhorende bode plot wordt bepaald. Het tweede signaal ecg2 heeft een lage en hoge frequentie ruis en deze wordt weggewerkt gebruikmakend van een FIR filter. Ten laatste wordt de sample frequentie van het signaal ecg verlaagt tot het laagste mogelijk niveau gebruikmakend van een IIR filter.

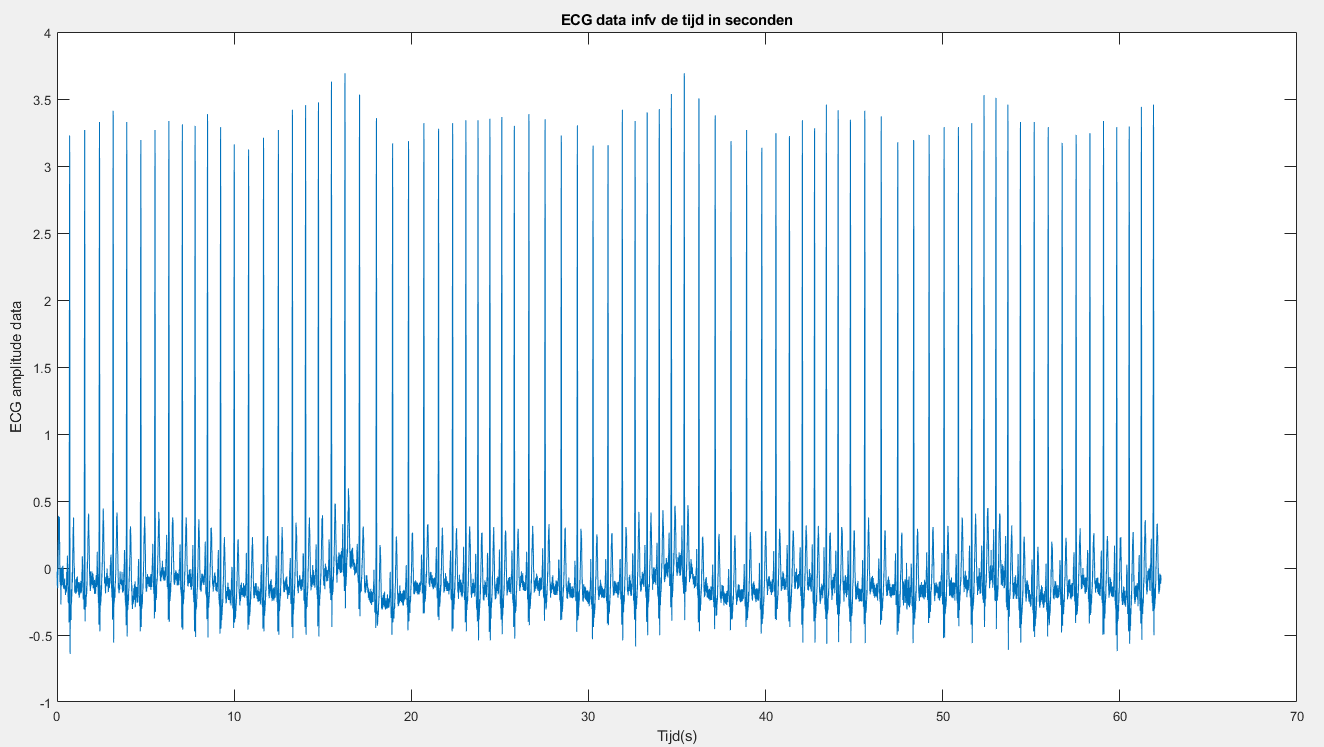
Enkele van de volgende afbeeldingen zijn lijngrafieken om de gegevens op een duidelijke manier te visualiseren, echter dit geeft de illusie dat de data van een continue bron komt. De gebruikte filters zijn echter digitaal en dus discreet. Deze vorm van visualisatie wordt gebruikt omdat als dit juist zou worden voorgesteld, met een discrete grafiek op figuur 2, de punten elkaar zo overlappen dat er enkel een balk van kleur zichtbaar is.

Task 1

De gegeven data bestaat uit de amplitude data van de signalen en de bijhorende sample frequentie . Om deze data te visualiseren wordt er eerst een kolom met tijdswaarden aan de data toegevoegd. De tijdswaarde van een bepaald datapunt op positie Q in de datalijst wordt berekend door Q te vermenigvuldigen met de sample periode . Dit resulteert in Figuur 1 en Figuur 2:



Figuur 1: Visualisatie van de ecg.mat dataset



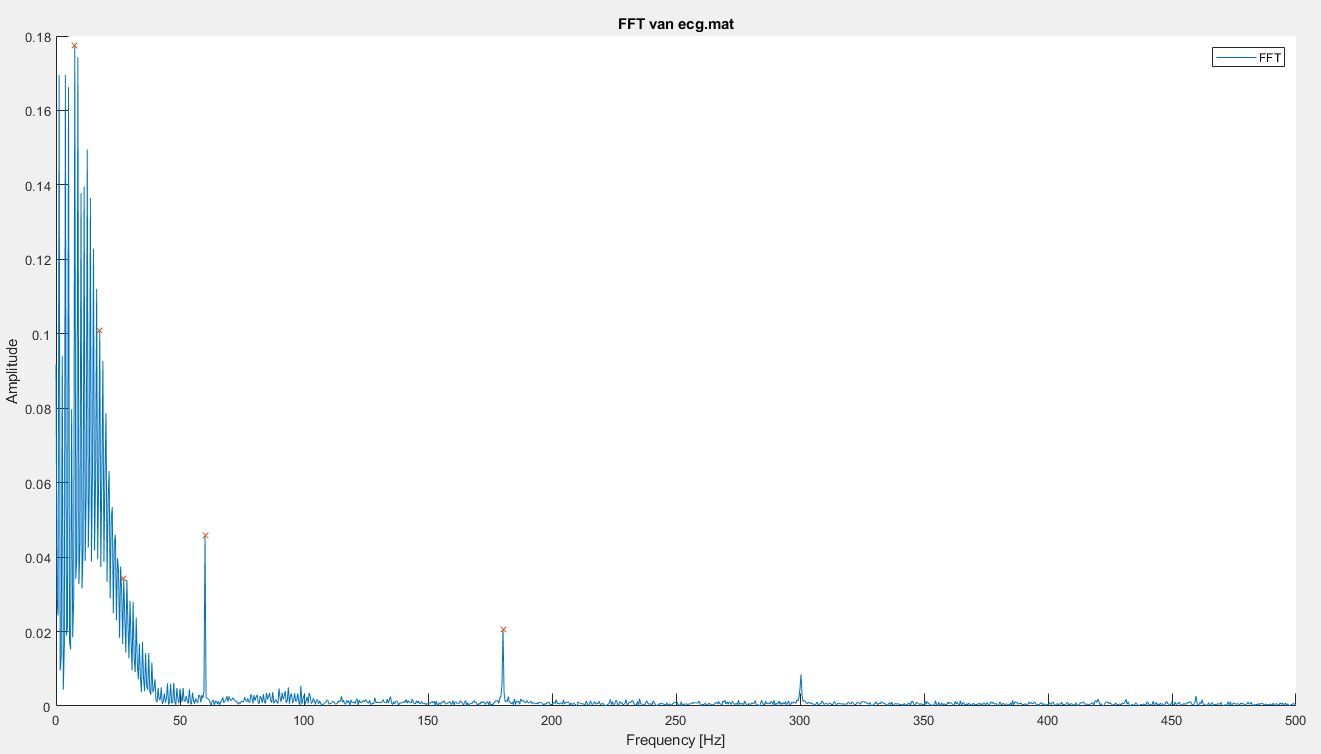
Figuur 2: Visualisatie van de ecg2.mat dataset

Task 2

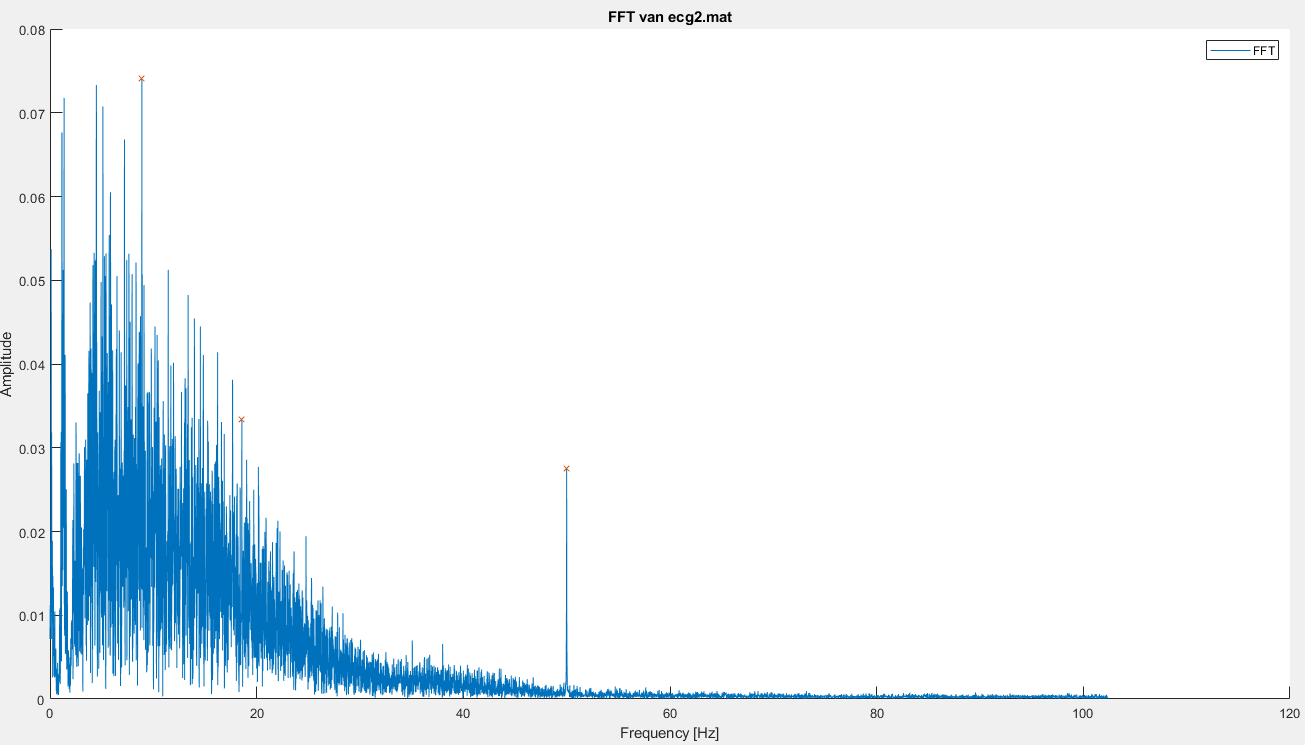
Power-line noise is ruis veroorzaakt door kabels waar wisselspanning van het elektriciteitsnet op staat. De oscillatie van de wisselspanning genereert een wisselend elektromagnetisch veld rond de kabel en het is dit veld dat kleine stromen, en dus ruis, in de meetapparatuur induceert. Dit betekent dat power-line noise een frequentie heeft van 50Hz, in Europa, of 60Hz, in de Verenigde Staten.

Om te bepalen op welke frequentie de power-line noise voorkomt wordt er van de signalen een Fast Fourier Transform (FFT) opgesteld. In de code wordt dit gedaan in de persoonlijke functie calculate\_FFT(Dataset, hoeveelheid datapunten in dataset, sample frequentie).

In de FFT grafieken, Figuur 3 en Figuur 4, is dan een piek te zien bij 50 Hz, voor de ecg2.mat dataset, en een piek bij 60 Hz voor de ecg.mat dataset. Voor het verwijderen van de power-line noise worden deze frequenties nauwkeuriger bepaald op de volgende wijze



Figuur 3: FFT van de ecg.mat dataset



Figuur 4: FFT van ecg2.mat dataset

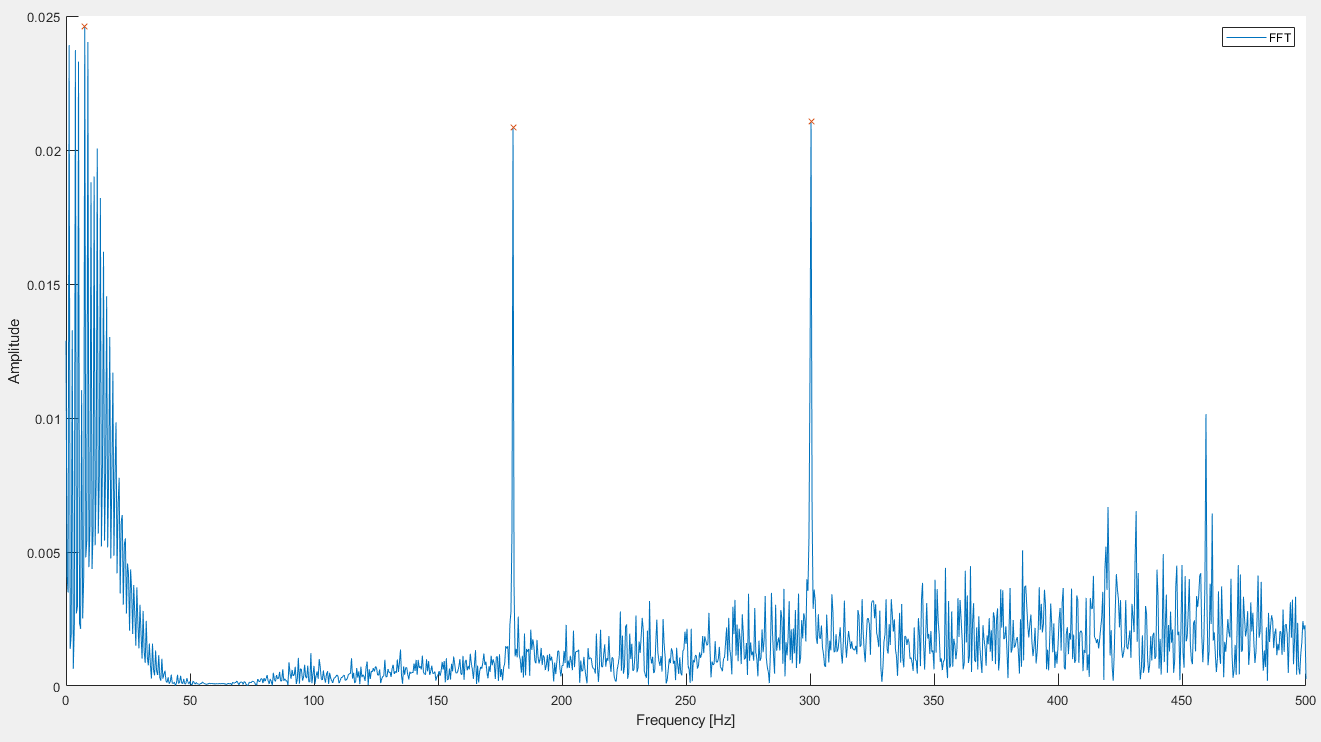
Uit deze grafieken kan ook worden afgeleid waar in het frequentie bereik de data zich bevindt. Voor ecg.mat bevat de data geen signalen met een frequentie hoger dan 50 Hz en voor ecg2.mat heeft de data geen frequentie elementen met een frequentie hoger dan 40 Hz. Deze waarden zullen van belang zijn bij het verwijderen van de hoog en laagfrequente ruis met een FIR filter en het verlagen van de sample frequentie voor de ecg.mat dataset.

Nu de frequentie van de power-line noise bepaald is kan deze worden verwijdert met de Notch filter. De Notch filter vergelijking bevat de en variabelen. Deze variabelen moeten zo worden berekent dat de juiste frequentie wordt uitgefilterd.

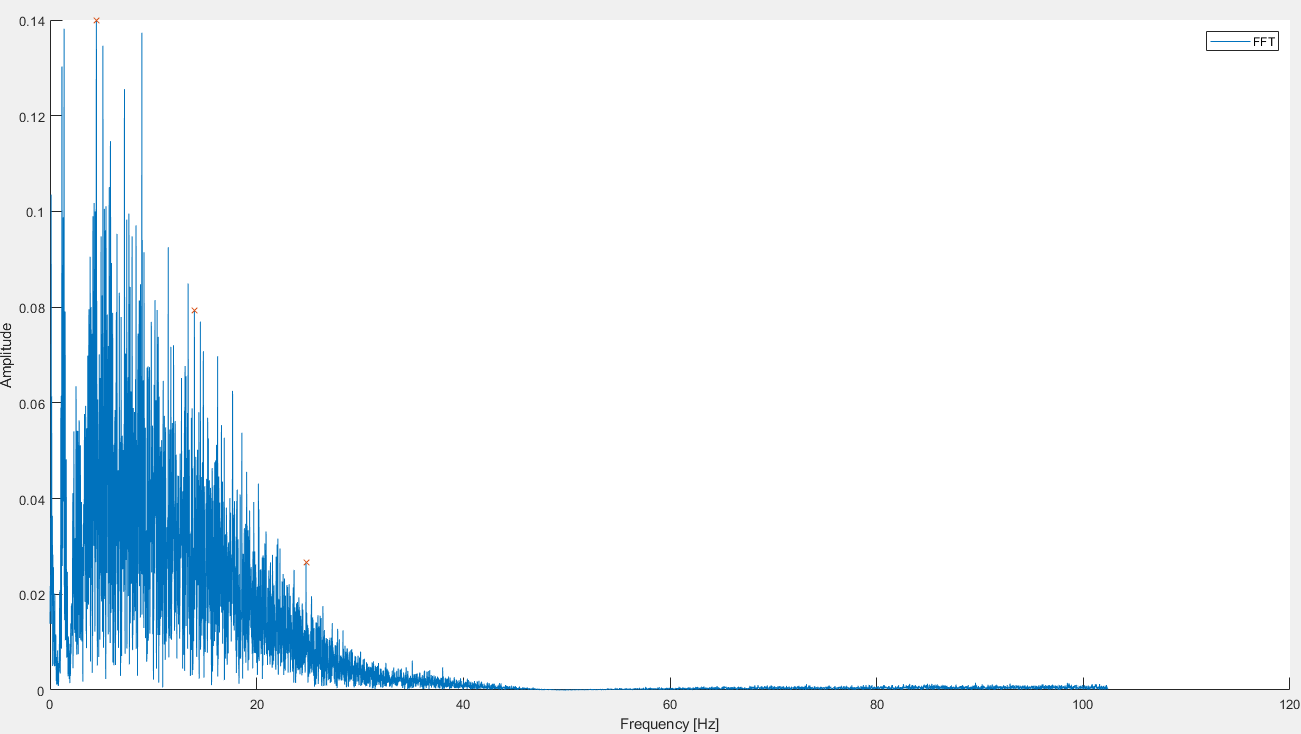
Na het berekenen van en wordt de data gefilterd. Dit gebeurt door de differentievergelijking van de notch filter op te stellen, zoals uitgebreid wordt gedaan in Task 3. Deze differentievergelijking wordt dan herschreven naar de uitgang:

Deze vergelijking kan rechtstreeks worden gebruikt om de data te filteren.

Van de gefilterde data “Notch” wordt nogmaals de FFT opgesteld om te zien of de het filteren juist gebeurt is zoals getoond in Figuur 5 en Figuur 6.

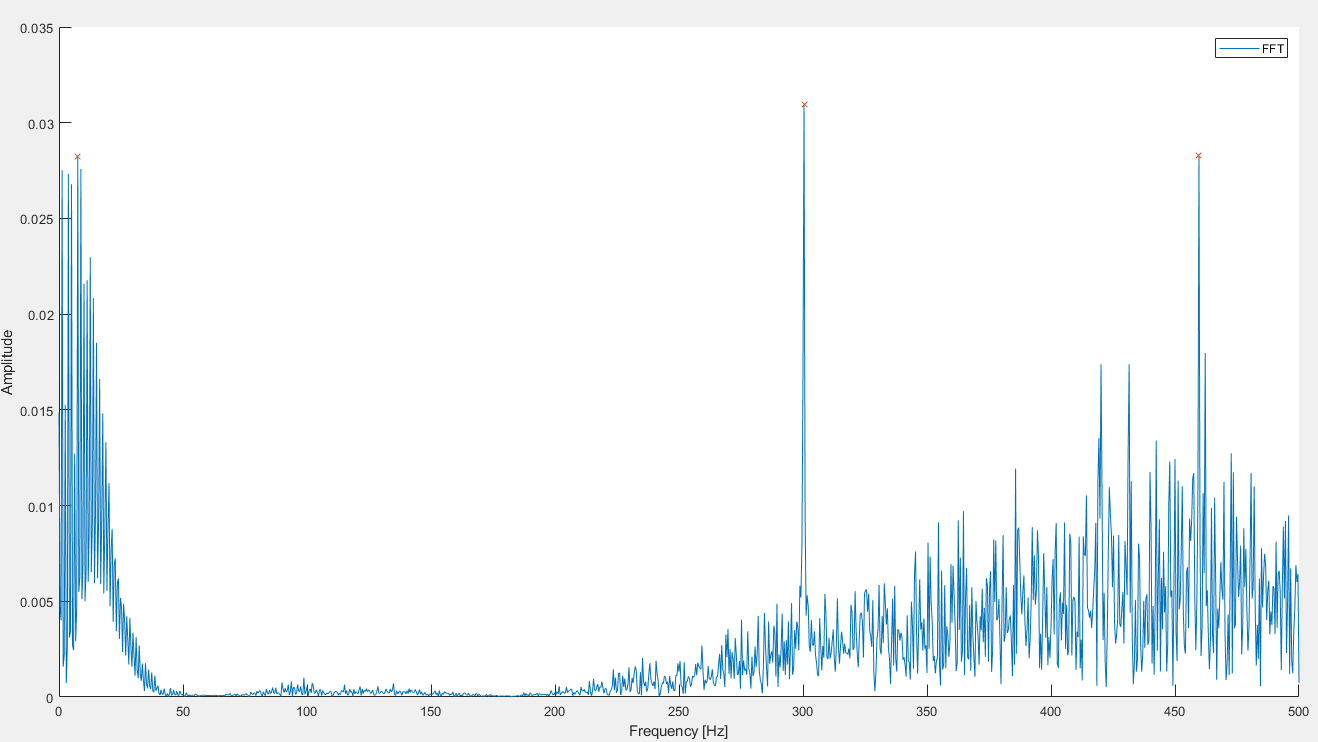


Figuur 5: FFT ecg.mat dataset na eenmalig te filteren met de Notch filter.

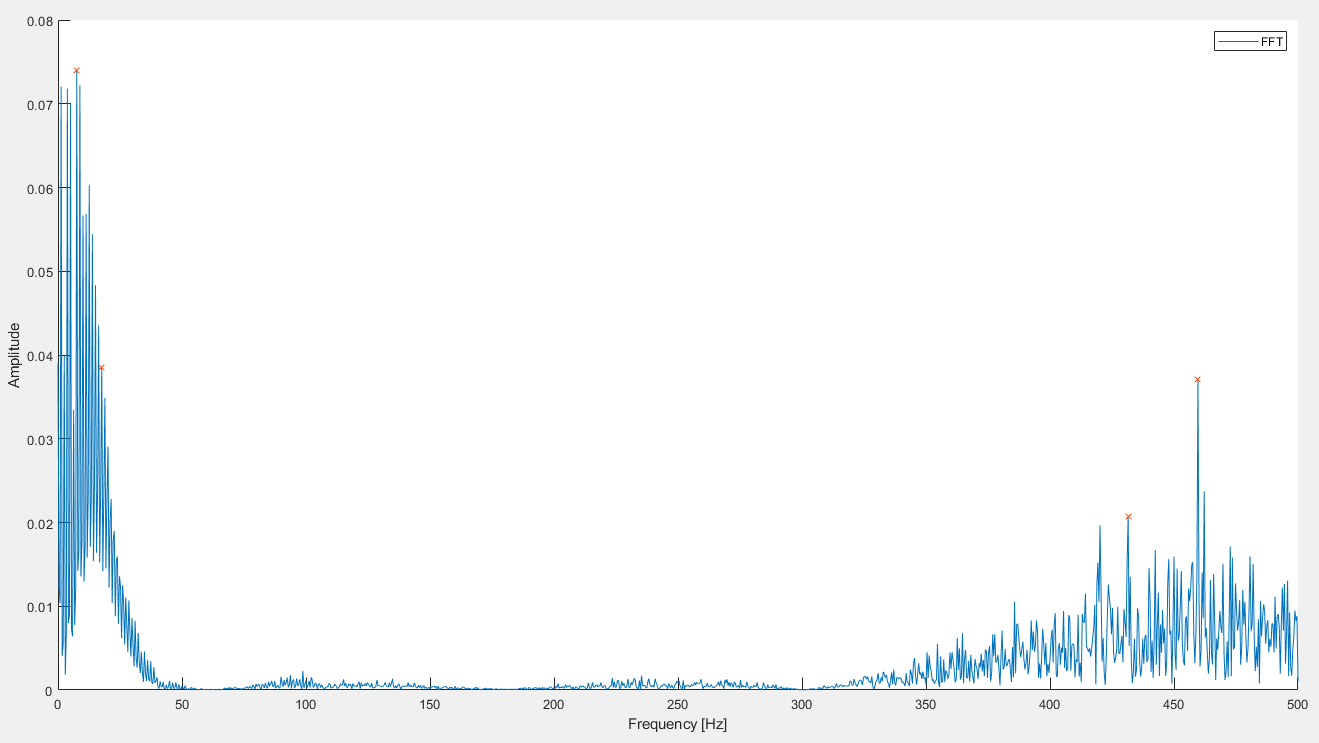


Figuur 6: FFT ecg2.mat dataset na eenmaal te filteren met de Notch filter.

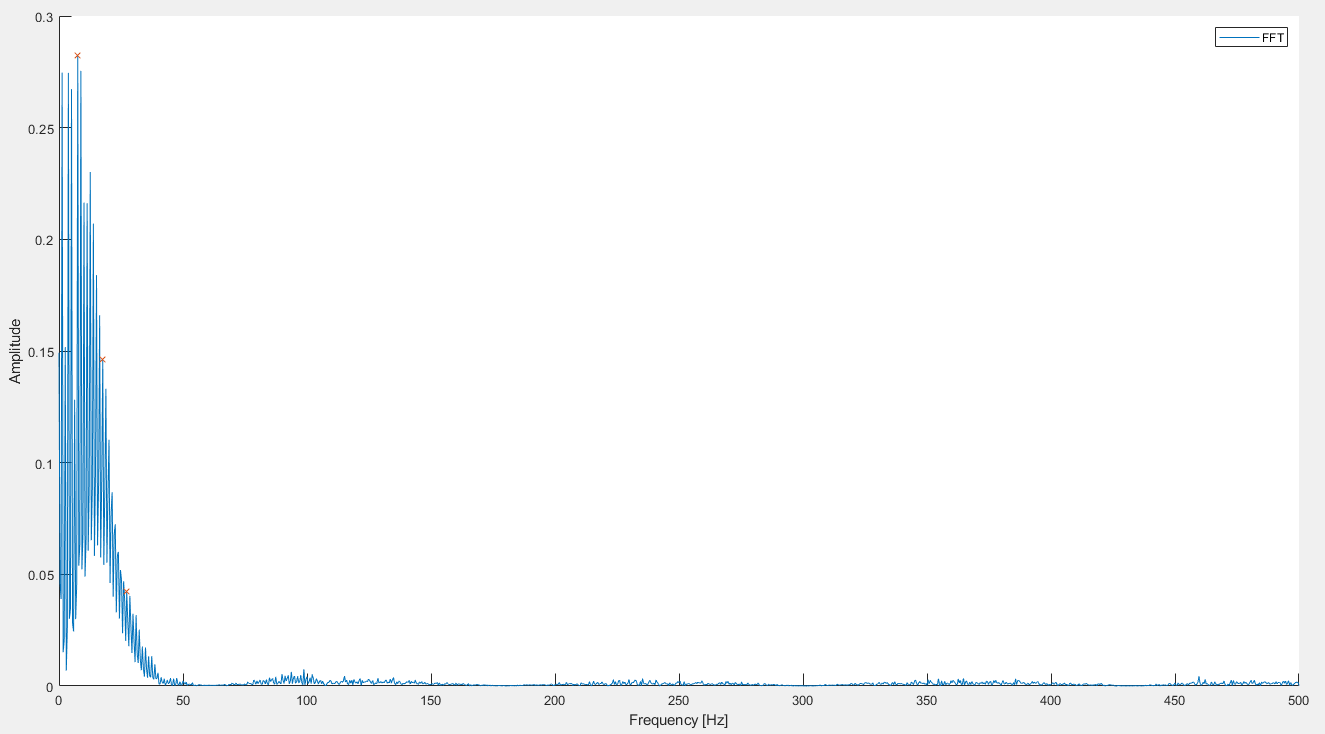
Voor ecg2.mat is eenmalig filteren voldoende om de power-line noise uit te filteren. Voor ecg.mat voegt de Notch filter extra ruis toe. Deze wordt ook weg gefilterd door meermaals een Notch filter op de data toe te passen maar steeds rond een hogere frequentie tot er geen ongewenste pieken meer voorkomen. Dit wordt getoond in Figuur 7, Figuur 8 en Figuur 9.



Figuur 7: ecg.mat na twee keer door de  Notch filter gefilterd te zijn.



Figuur 8: ecg.mat na drie keer door de  Notch filter gefilterd te zijn.



Figuur 9: ecg.mat na vier keer door de  Notch filter gefilterd te zijn.

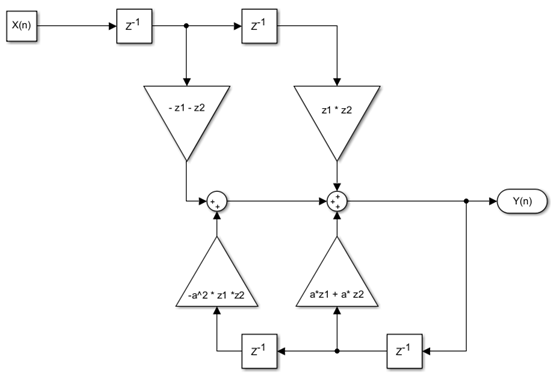
Zowel bij ecg.mat en ecg2.mat blijft er nog hoogfrequente ruis over. Deze zal worden verwijdert als onderdeel van Task 6 en Task 7.

Task 3

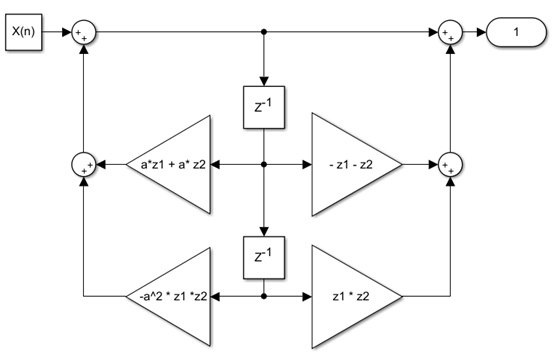
Het filteren van de data gebeurt door, zoals eerder vermeld, gebruik te maken van de differentievergelijking toe te passen op de dataset. Deze differentievergelijking wordt opgesteld door de transferfunctie op de volgende wijze te herschrijven:

Deze differentievergelijking kan ook worden gebruikt om een visuele representatie te geven van de filter. Deze representatie kan op twee manieren gebeuren, volgens de Direct Form I en de Direct Form II.

De Direct Form 1, te zien in Figuur 10, Is de implementatie waar minder rekenkracht voor nodig is. Deze implementatie heeft echter een grotere hoeveelheid vertraging elementen nodig. De Direct Form II is dan weer economischer met het geheugen maar hiervoor zijn meer berekeningen nodig. De implementatie van de differentievergelijking voor Task 2 is volgens de Direct Form II.



Figuur 10: Direct Form I representatie van de Notch Filter



Figuur 11: Direct Form II representatie van de Notch Filter

Task 4

Voor het bepalen van de impulsrespons en de frequentierespons werd dit eerst met de hand gedaan en daarna controleert met de Matlab functies impz en freqz. De impulsrespons bepaalt men door gebruik te maken van de differentievergelijking: .

De impulsrespons is dan een Dirac-impuls (en ) ingevuld in de differentievergelijking en voor iedere stap berekend:

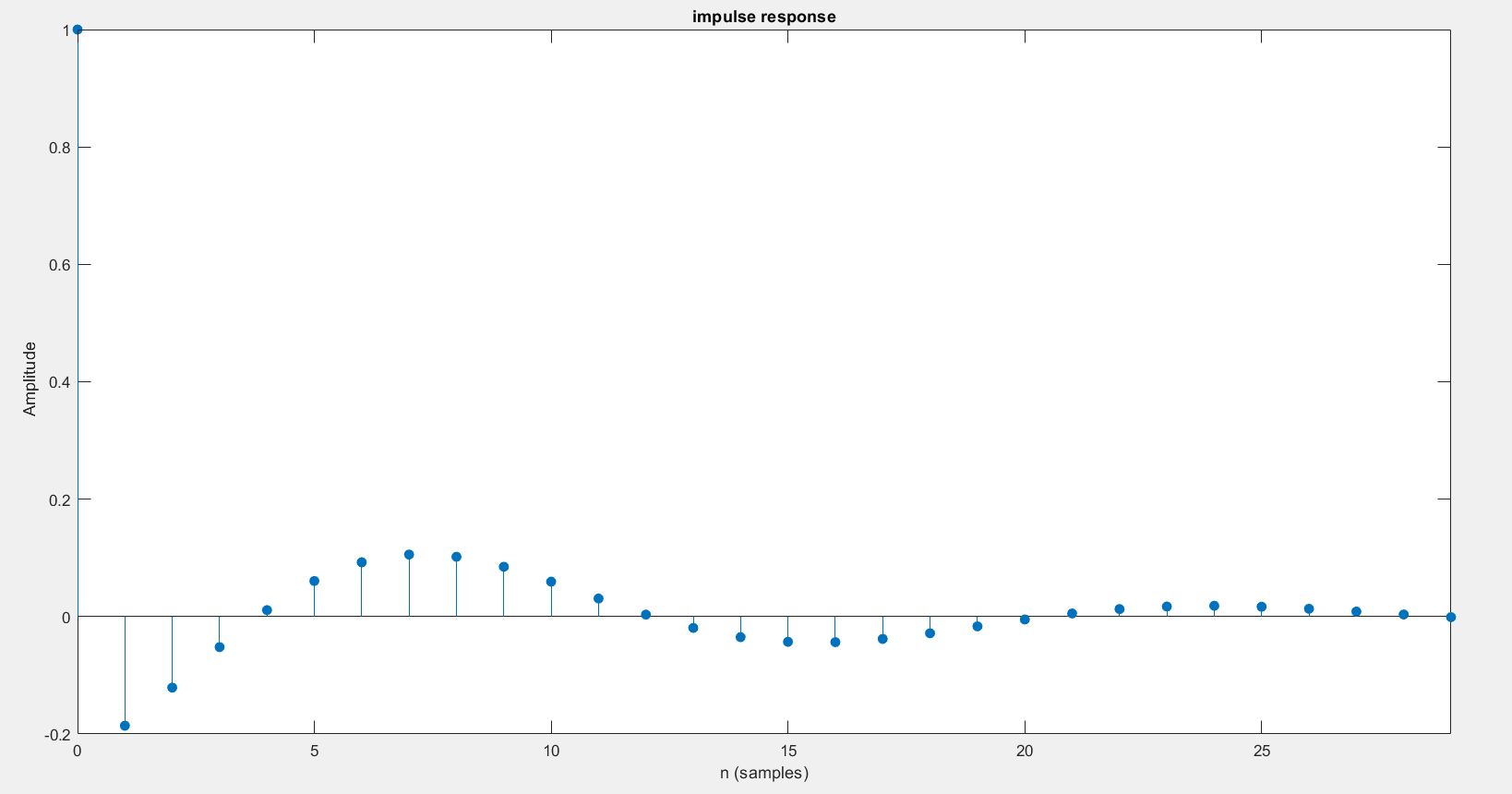
Met

De frequentierespons is opgedeeld in 2 delen, de magnitude van de respons en de faseverschuiving van de respons. Deze worden bepaald voor de frequentie op , en . Om de magnitude te bepalen moet men in vullen in de transfert functie en de absolute waarde hiervan bepalen. De faseverschuiving wordt bepaald door de boogtangens te bepalen van het imaginair gedeelde over het reëel gedeelde.

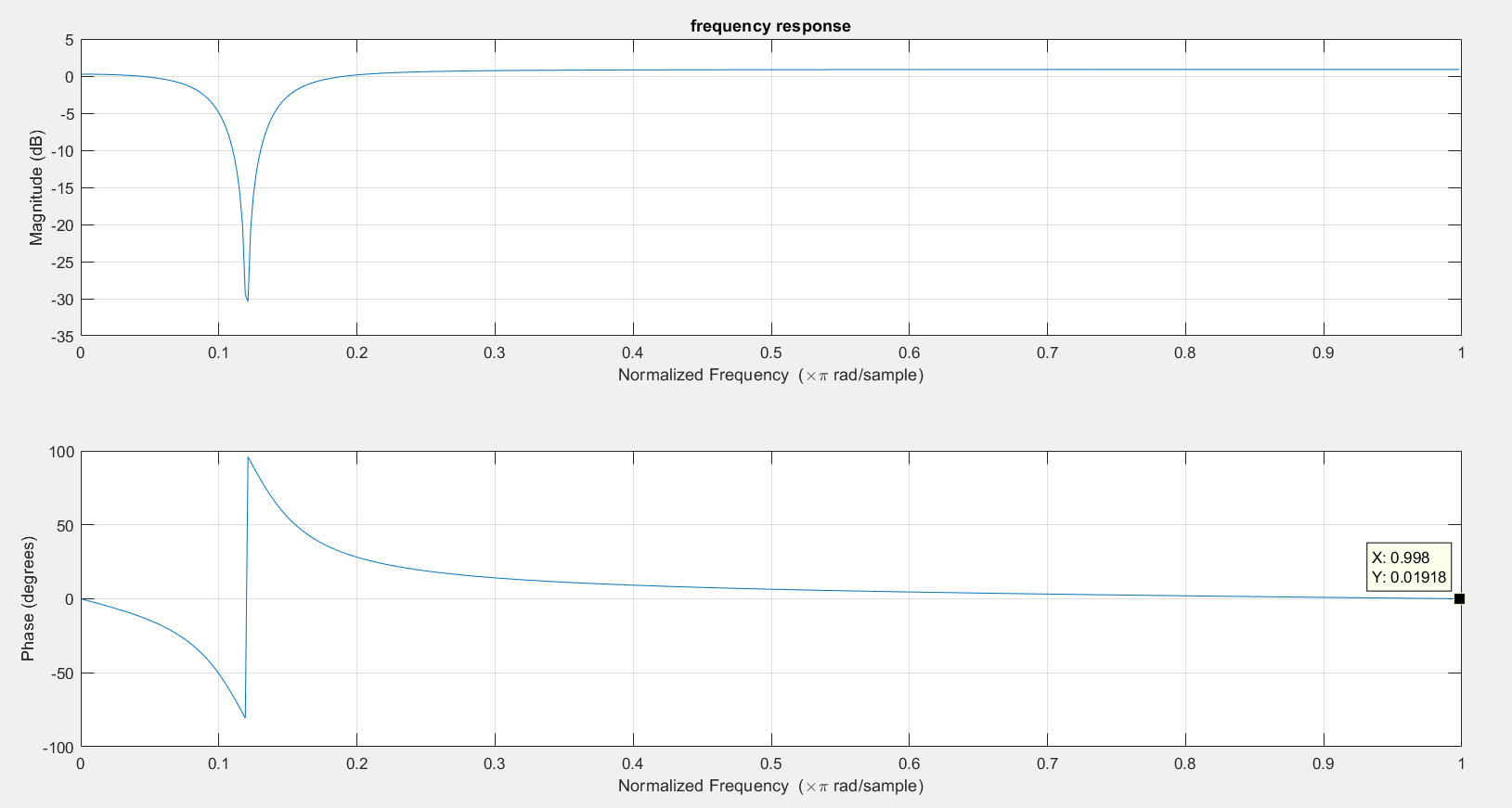
Met en

Voor de magnitude in decibel wordt dit . Invullen voor de frequentie op , en geeft voor respectievelijk en . De magnitude wordt dan . Voor en wordt de magnitude dan en respectievelijk. De faseverschuiving wordt dan bepaald door:

Invullen voor gelijk aan en geeft respectievelijk een faseverschuiving van 0°, 6.477° en 0°. Figuur 12 en 13 tonen de berekende waarden voor de impulsrespons en de frequentierespons. Hieruit is duidelijk dat onze berekende waarden overeenkomen met de waarde bepaald door Matlab.



Figuur 12: Impulsrespons van Matlab functie impz

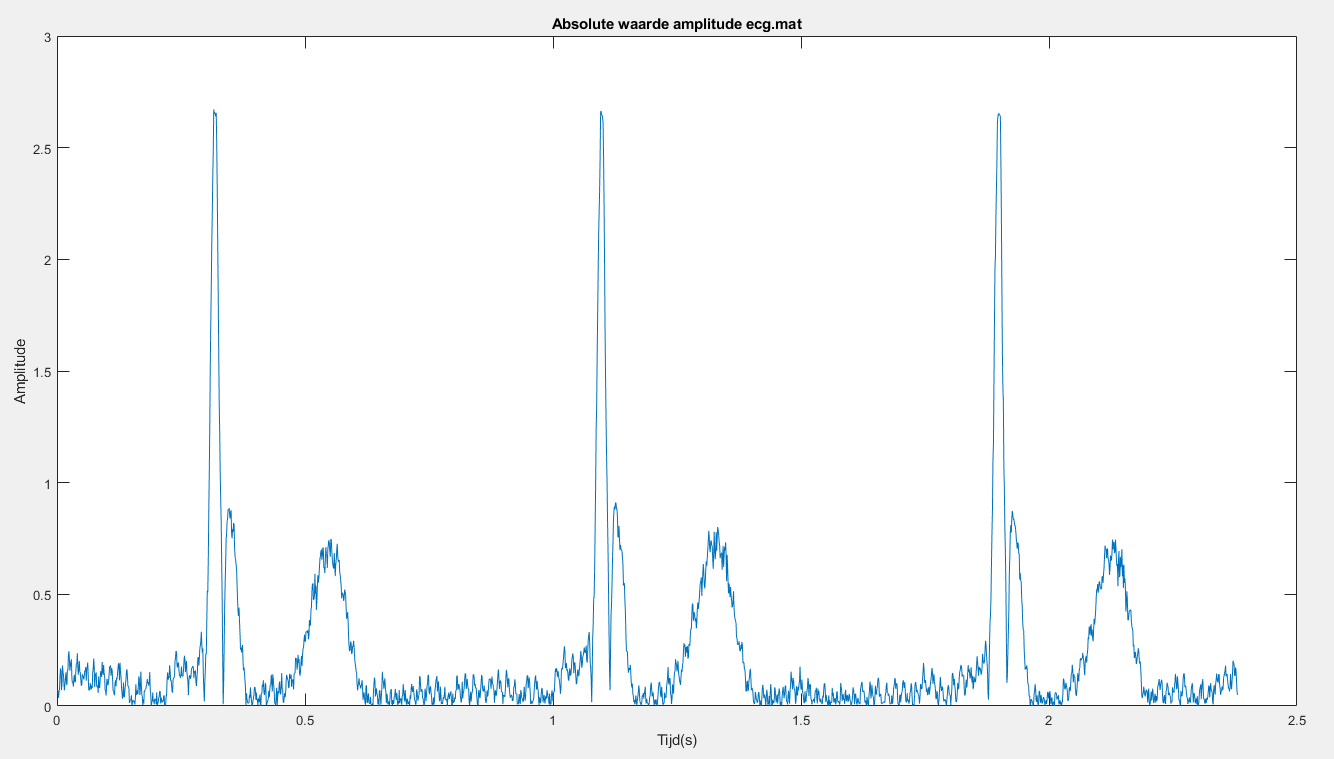


Figuur 13: Frequentierespons van Matlab functie freqz

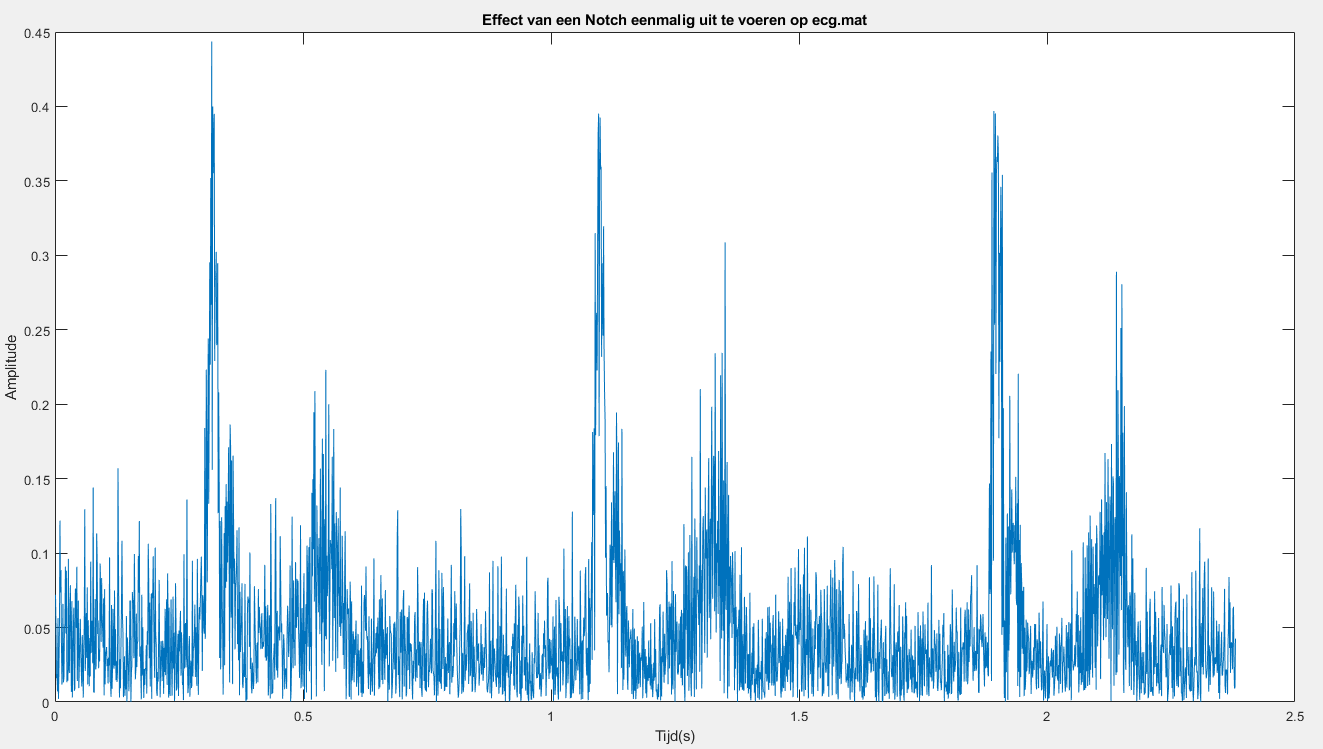
Task 5

In Task 2 is het frequentiegedrag van de gefilterde signalen al bestudeerd. De Figuren 3 tot en met 9 tonen het frequentiespectrum van de ecg signalen in verschillende stadia van het filter proces.

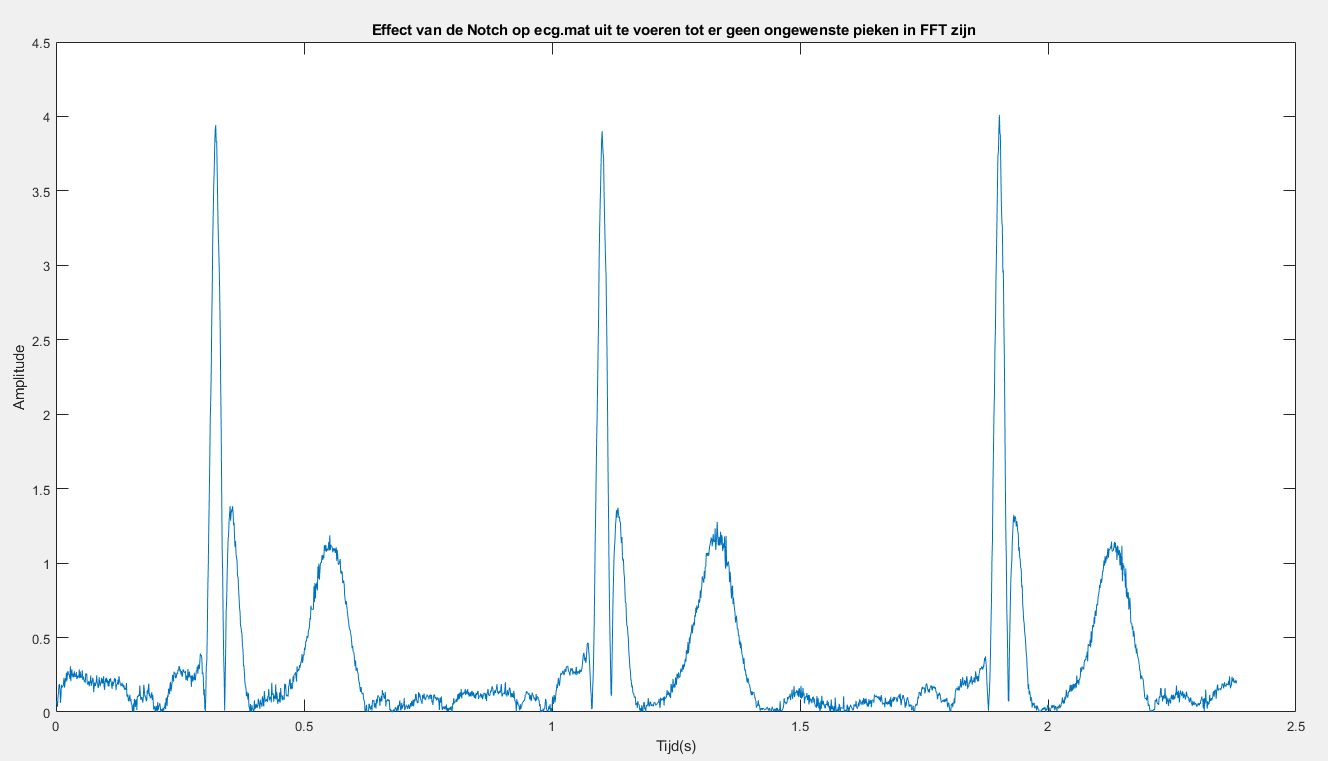
De originele signalen, zie Figuur 1 en 2, worden door het filter proces aangepast. Om een goede vergelijking te maken worden de amplitudes van ecg.mat en ecg2.mat gekwadrateerd waarna er de wortel van wordt genomen. Dit wordt gedaan omdat na het filteren de amplitudes complexe elementen krijgen. Deze complexe componenten zijn weg te werken tot amplitudes maar enkel met positieve waarden aan de hand van



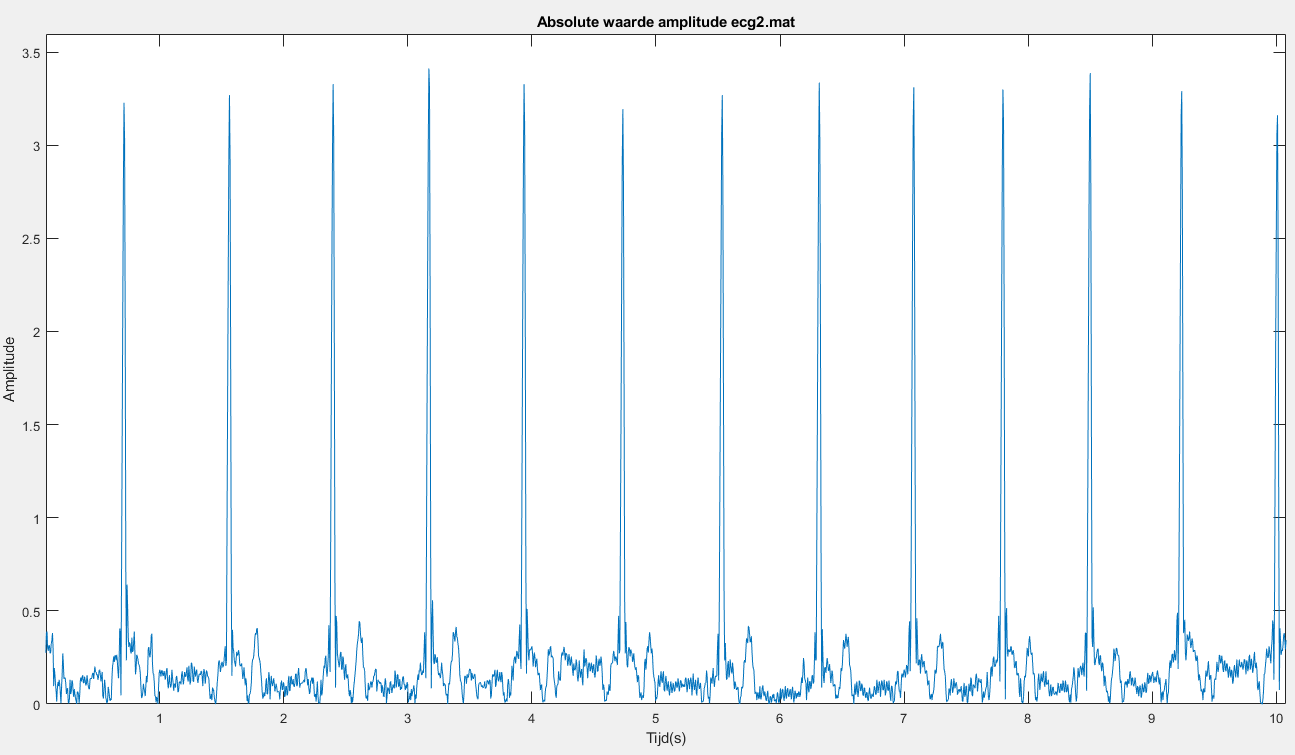
Figuur 14: Absolute waarde van de amplitudes in ecg.mat



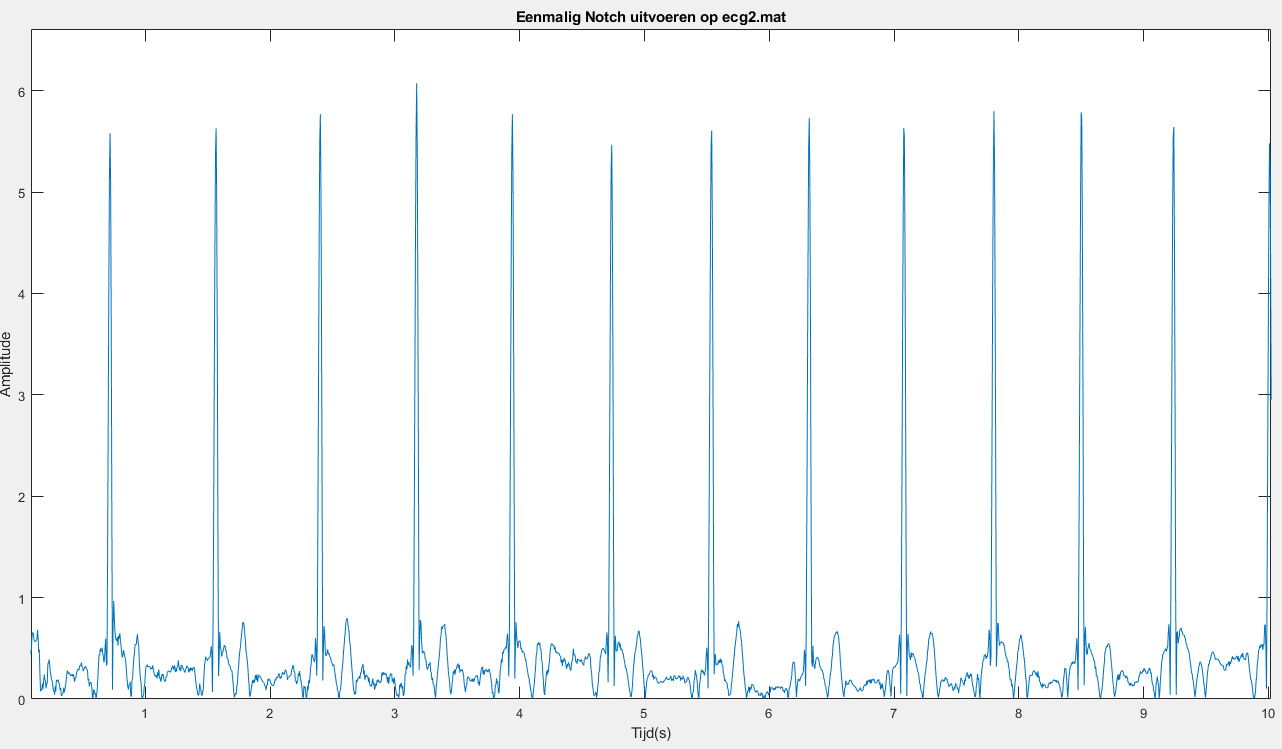
Figuur 15: Effect van het eenmalig uitvoeren van de Notch filter op ecg.mat



Figuur 16: Absolute amplitude waarden behorende bij FFT uit Figuur 9



Figuur 17: Absolute waarde van de amplitudes in ecg2.mat voor second 0 tot 10

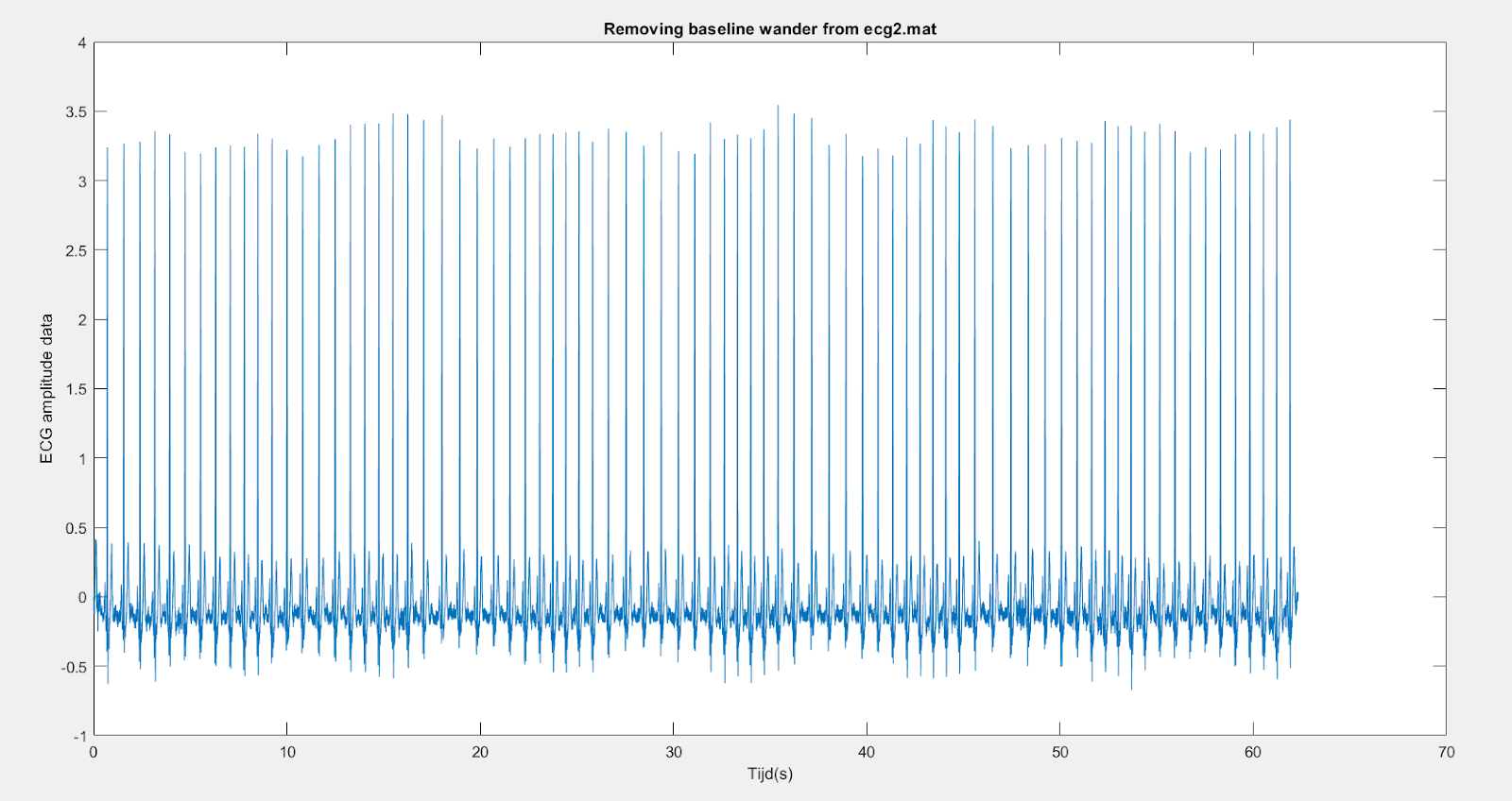


Figuur 18: Absolute amplitude waarden behorende bij FFT uit Figuur 6 voor second 0 tot 10

Uit het verschil tussen Figuur 14 en Figuur 16 is het effect van power-line noise duidelijk te zien. Dit is ook duidelijk bij het verschil tussen Figuur 17 en Figuur 18. Hieruit valt te concluderen dat het filter proces beschreven in Task 2 naar behoren functioneert.

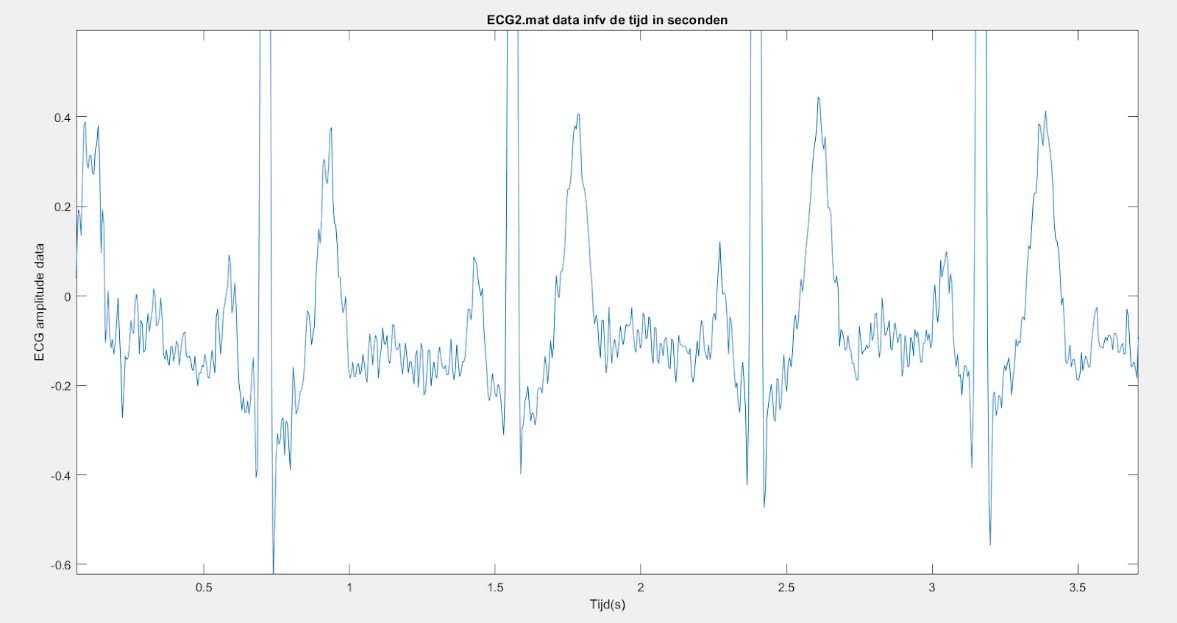
Task 6

Door ademhaling en lichamelijke bewegingen ontstaat een laagfrequente ruis op het ECG signaal. Deze ruis veroorzaakt een drift waardoor het signaal van zijn basislijn afwijkt. Hierdoor is moeilijker om de pieken in het signaal te detecteren en analyseren, zoals te zien in figuur 2. Deze drift wordt weg gefilterd met behulp van een FIR high pass filter met een cutoff frequentie van 0.5 Hz. Het resultaat in Figuur 19 heeft duidelijk minder laagfrequente verplaatsingen.

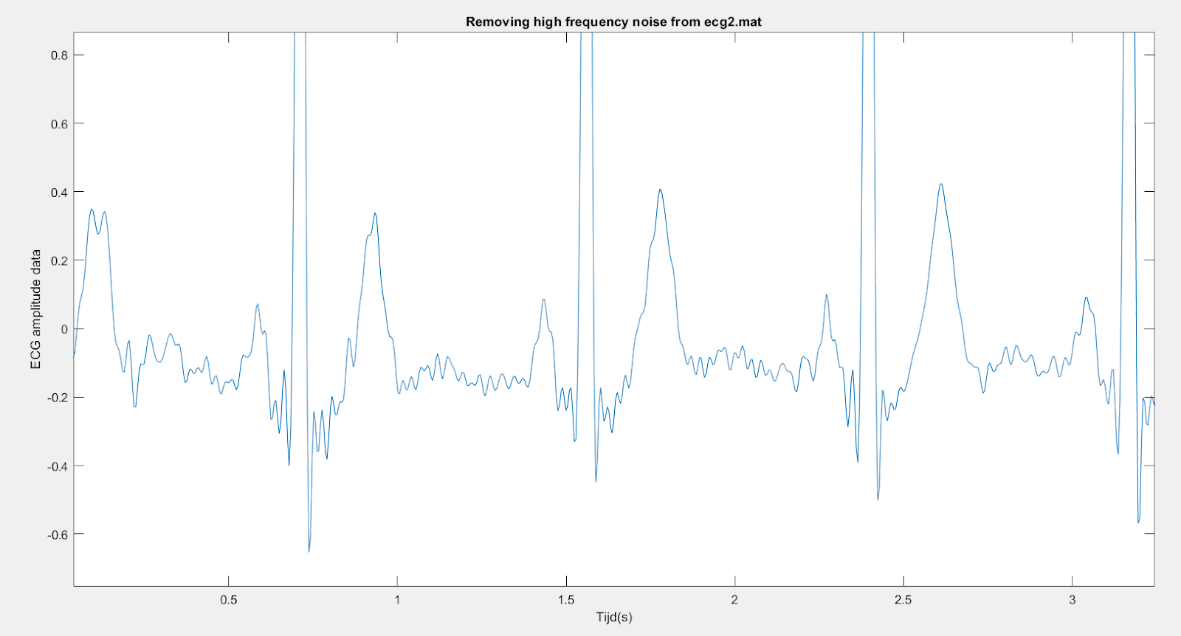


Figuur 19: Laagfrequente baseline verplaatsing verwijderd met high pass filter

Ook ontstaat er hoog frequent ruis op het ECG signaal. Om deze weg te filteren wordt een FIR low pass filter gemaakt met cutoff frequentie van 40 Hz om het hoogfrequente ruis weg te filteren. Figuur 20 toont het hoogfrequente ruis op het ecg2.mat signaal en figuur 19 toont het gefilterd signaal.



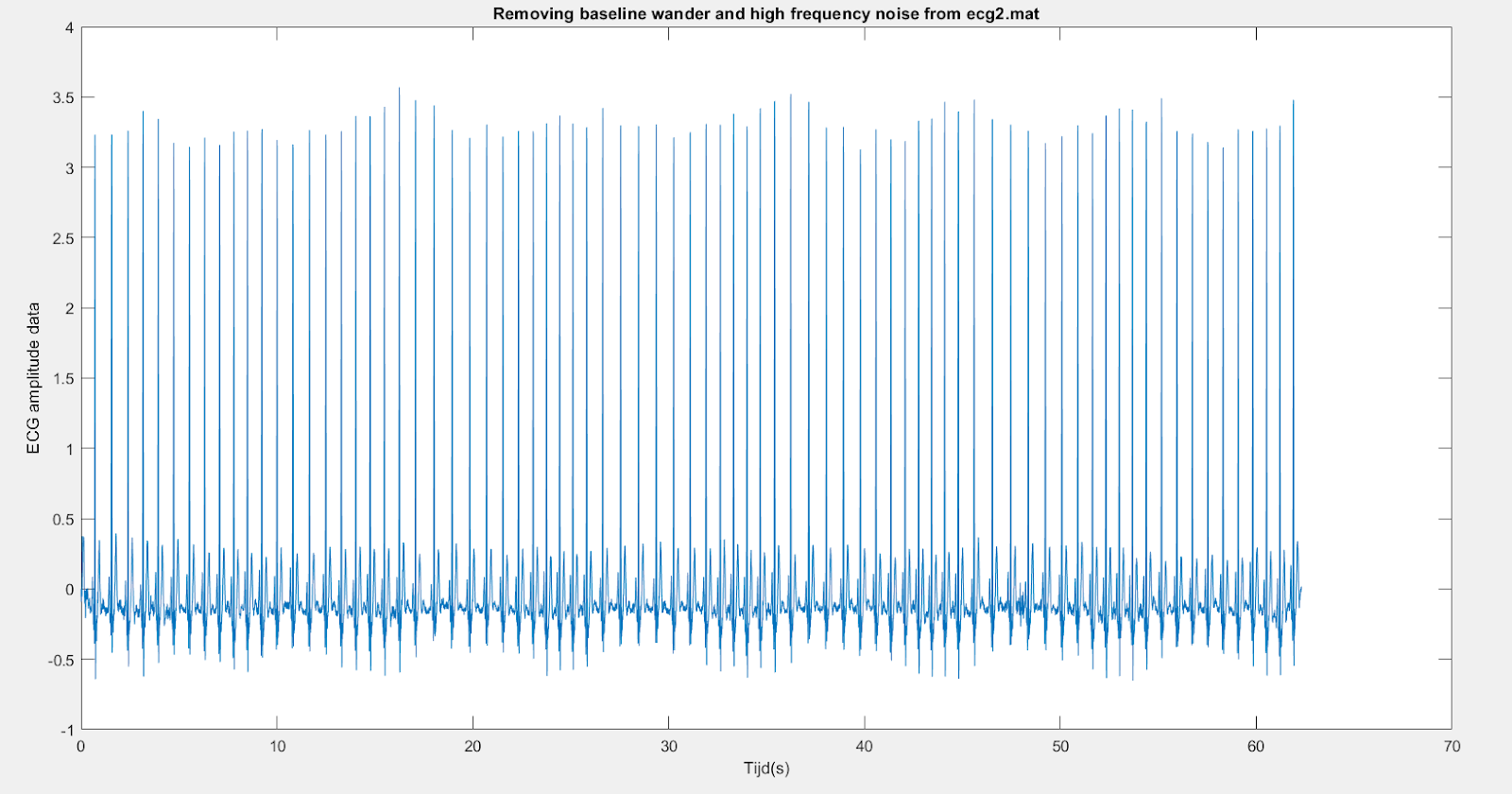
Figuur 20: Hoog frequent ruis op ecg2.mat



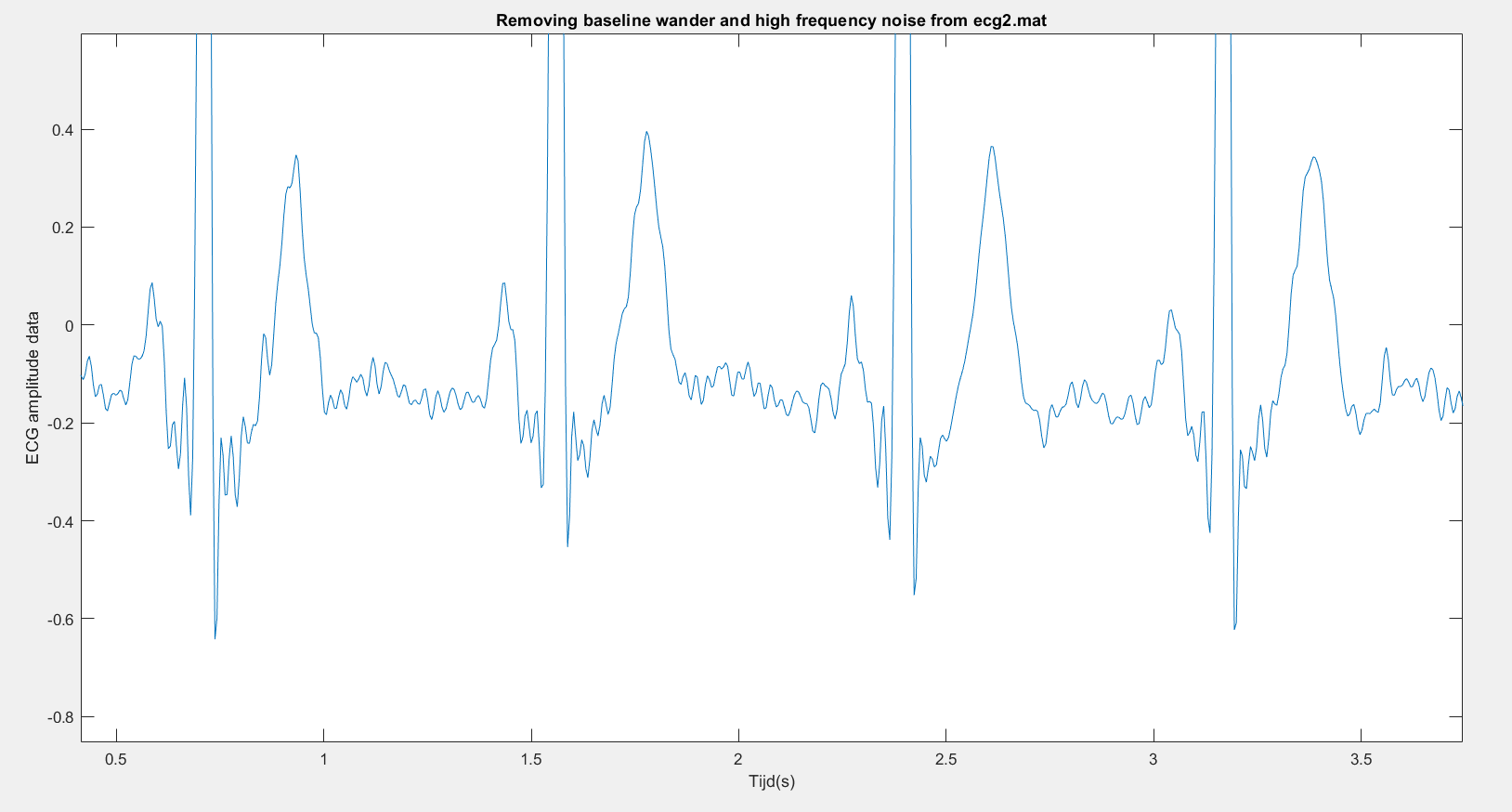
Figuur 21: Hoogfrequente ruis verwijderd met low pass filter

Duidelijk is er een vermindering van ruis op de signalen.

Om het hoogfrequente en laagfrequente ruis tegelijkertijd weg te werken, wordt gebruik gemaakt van een FIR band pass filter. Het resultaat wordt getoond in Figuur 21.



Figuur 22: Verwijderen van baseline verplaatsing met bandpass



Figuur 23: Verwijderen van hoogfrequent ruis met bandpass

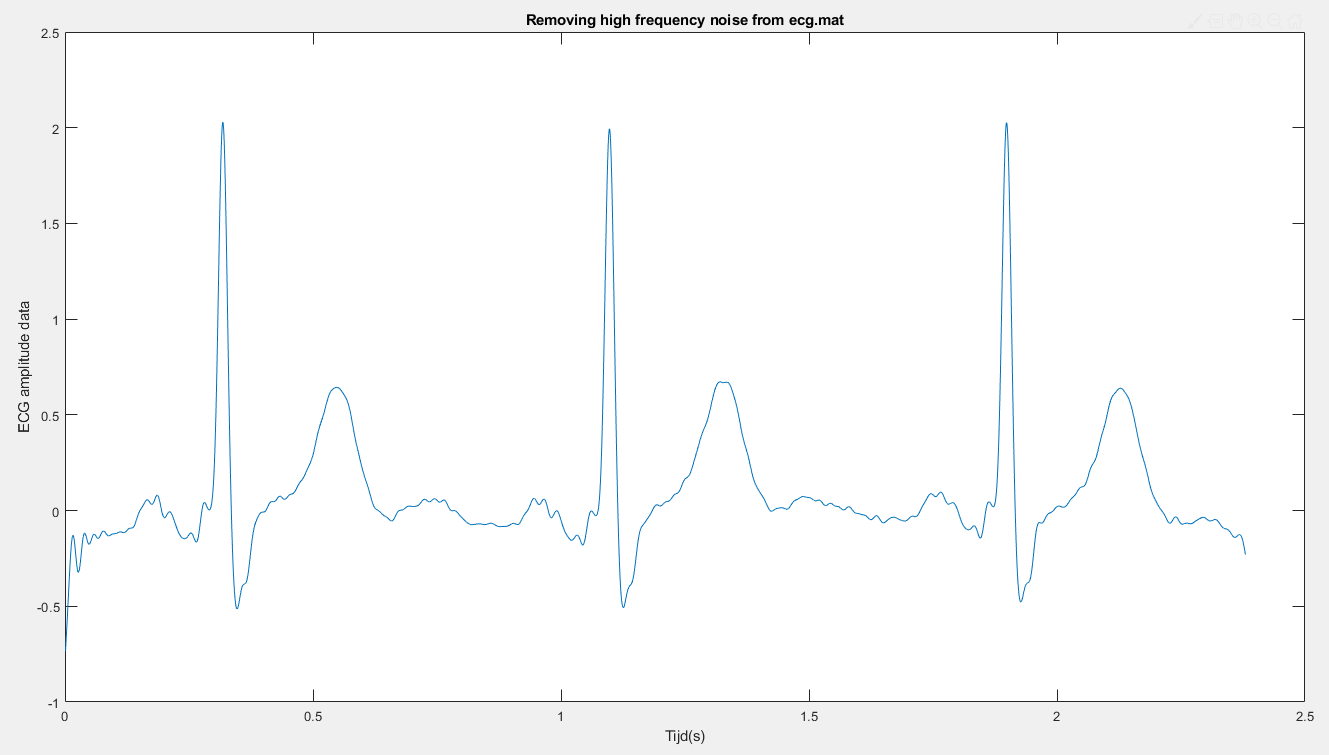
Figuur 22 en 23 tonen aan dat het verwijderen van de baseline verplaatsing en de hoogfrequente ruis in 1 stap mogelijk is door gebruik van een bandpass filter.

Task 7

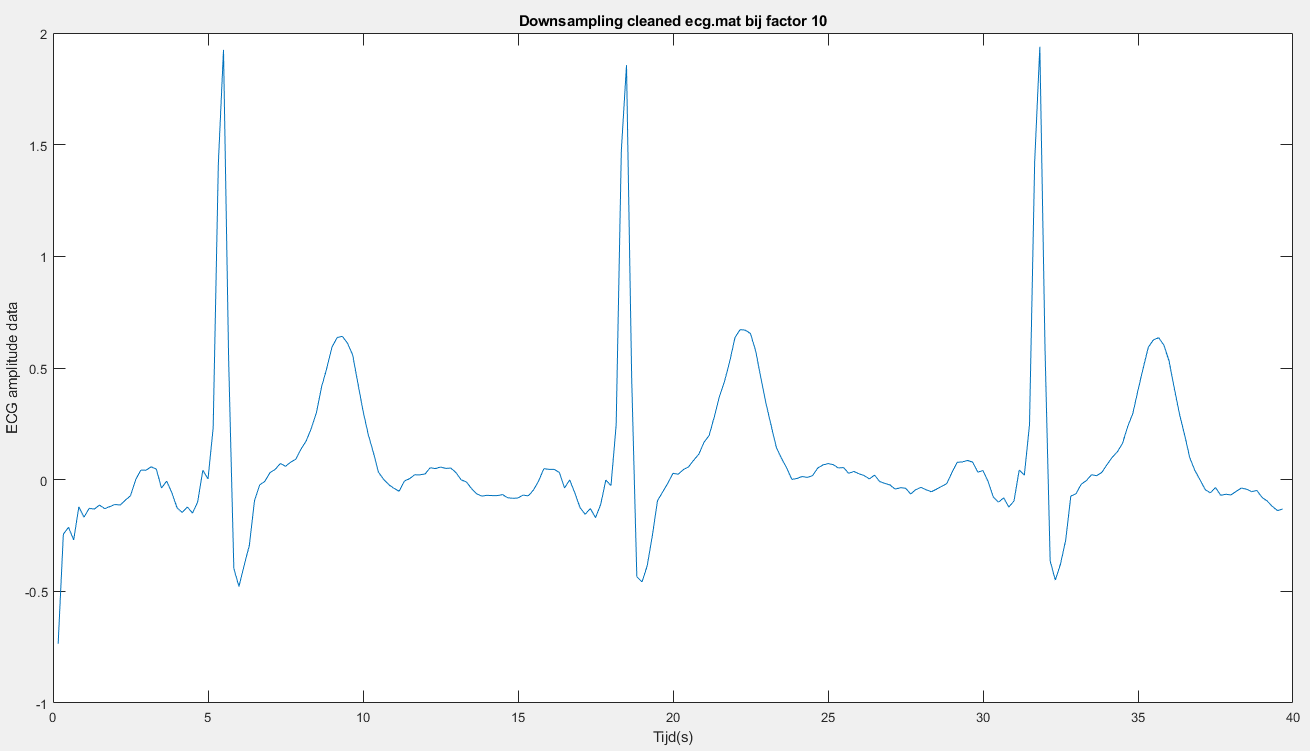
Het ecg.mat signaal is gemeten met een samplefrequentie van 1kHz. Dit is echter veel te hoog voor de frequenties van de data die wordt gemeten. Hierdoor wordt er te veel geheugen gebruikt voor het opslaan van de data. Hierom is het voordelig om de samplefrequentie te verlagen en meer geheugen vrij te maken.

Voordat de samplefrequentie kan worden verlaagt moet eerst alle ruis met een frequentie groter dan de helft van de nieuwe gewenste samplefrequentie worden weg gefilterd om aliasing op het uiteindelijke signaal te voorkomen. Dit wordt gedaan met een laagdoorlaatfilter met een cut-of frequentie van 55 Hz. Deze frequentie wordt gekozen door uit Figuur 9 de hoogste data dragende frequentie te bepalen.

Na het uitfilteren van de hoogfrequente ruis kan de sample frequentie worden verlaagt door van iedere , ongeveer 10, samples enkel het eerste datapunt bij te houden. Als dit correct gebeurt zou er zo goed als geen verschil mogen zijn tussen het signaal in Figuur 24, en het gedownsampled signaal in Figuur 25. Duidelijk is dat het gedownsampled signaal niet afwijkt van het oorspronkelijk.



Figuur 24: Ecg.mat signaal na LP filtering



Figuur 25: Ecg.mat signaal na LP filtering en downsampling

De Matlab Code

%% clear workspace and commandline and close all figures

clc;

clear;

close all;

%% load provided data

load("ecg.mat");

load("ecg2.mat");

measure\_freq\_ecg = 1000; % Hz

measure\_freq\_ecg2 = 204.73; % Hz

%% get amount ecg data points

[ecg\_rows, ~] = size(ecg);

[ecg2\_rows, ~] = size(ecg2);

%% Task 1:

%visualize sample data:

% add a column with time values to ecg data

ecg\_timecolumn = [ones(ecg\_rows, 1), ecg];

ecg\_timecolumn = convert\_to\_time\_and\_plot(ecg\_timecolumn, measure\_freq\_ecg, ecg\_rows);

title("ECG.mat data infv de tijd in seconden");

ecg2\_timecolumn = [ones(ecg2\_rows, 1), ecg2];

ecg2\_timecolumn = convert\_to\_time\_and\_plot(ecg2\_timecolumn, measure\_freq\_ecg2, ecg2\_rows);

title("ECG2.mat data infv de tijd in seconden");

pause;

%% Task 2:

% powerline spectrum:

%visualization

[FFT\_amp\_ecg\_0, freq\_ecg\_0] = calculate\_FFT(ecg, ecg\_rows, measure\_freq\_ecg);

title('FFT van ecg.mat')

[FFT\_amp\_ecg2\_0, freq\_ecg2\_0] = calculate\_FFT(ecg2, ecg2\_rows, measure\_freq\_ecg2);

title('FFT van ecg2.mat')

%find PL noise frequency ecg and ecg2

thresh\_0 = 44; %Hz 50 are 60 Hz are sources for the PLN

PLN\_freq\_degrees\_ecg\_0 = freq\_ecg\_0(find(freq\_ecg\_0 > thresh\_0,1));

PLN\_freq\_degrees\_ecg2\_0 = freq\_ecg2\_0(find(freq\_ecg2\_0 > thresh\_0,1));

%calculate notch and apply filters Transfer Functions:

a = 0.9;

[Notch\_ecg\_1, z1\_ecg\_1, z2\_ecg\_1] = calculate\_notch\_with\_conj(ecg\_timecolumn, PLN\_freq\_degrees\_ecg\_0, a, measure\_freq\_ecg, ecg\_rows);

[~, freq\_ecg\_1] = calculate\_FFT(Notch\_ecg\_1(:,2), ecg\_rows, measure\_freq\_ecg);

title('ecg.mat dataset na eenmalig te filteren met de Notch filter.')

thresh\_1 = 100;

[Notch\_ecg\_2, z1\_ecg\_2, z2\_ecg\_2] = clean\_notch(thresh\_1, freq\_ecg\_1, Notch\_ecg\_1, a, measure\_freq\_ecg, ecg\_rows);

[Notch\_ecg2\_1, z1\_ecg2, z2\_ecg2] = calculate\_notch\_with\_conj(ecg2\_timecolumn, PLN\_freq\_degrees\_ecg2\_0, a, measure\_freq\_ecg2, ecg2\_rows);

[~, ~] = calculate\_FFT(Notch\_ecg2\_1(:,2), ecg2\_rows, measure\_freq\_ecg2);

title('ecg2.mat dataset na eenmalig te filteren met de Notch filter.')

pause;

%% Task 3

% pen and paper, see report.

%% Task 4

radials = PLN\_freq\_degrees\_ecg\_0 \*2\*pi / 1000;

z1 = cos(radials) + 1j \* sin(radials);

z2 = conj(z1);

impulse\_response(z1 ,z2);

%% Task 5

%ecg.mat

figure

plot(ecg\_timecolumn(:,1), sqrt(ecg\_timecolumn(:,2).^2))

xlabel('Tijd(s)')

ylabel('Amplitude')

title('Absolute waarde amplitude ecg.mat')

%Na eenmaal Notch toe te passen

i = imag( Notch\_ecg\_1(:,2));

r = real( Notch\_ecg\_1(:,2));

a = sqrt(r.^2 + i.^2);

figure

plot(ecg\_timecolumn(:,1), a)

xlabel('Tijd(s)')

ylabel('Amplitude')

title('Effect van een Notch eenmalig uit te voeren op ecg.mat')

%Na het beindigen van het filter process

i = imag( Notch\_ecg\_2(:,2));

r = real( Notch\_ecg\_2(:,2));

a = sqrt(r.^2 + i.^2);

figure

plot(ecg\_timecolumn(:,1), a)

xlabel('Tijd(s)')

ylabel('Amplitude')

title('Effect van de Notch op ecg.mat uit te voeren tot er geen ongewenste pieken in FFT zijn ')

%ecg2.mat

figure

plot(ecg2\_timecolumn(:,1), sqrt(ecg2\_timecolumn(:,2).^2))

xlabel('Tijd(s)')

ylabel('Amplitude')

title('Absolute waarde amplitude ecg2.mat')

%Na eenmaal Notch toe te passen

i = imag( Notch\_ecg2\_1(:,2));

r = real( Notch\_ecg2\_1(:,2));

a = sqrt(r.^2 + i.^2);

figure

plot(ecg2\_timecolumn(:,1), a)

xlabel('Tijd(s)')

ylabel('Amplitude')

title('Eenmalig Notch uitvoeren op ecg2.mat')

%% Task 6

%removing baseline wander

HP\_filter = designfilt('highpassfir',... % Response type

'StopbandFrequency',0.3, ... % Frequency constraints

'PassbandFrequency',0.5, ...

'StopbandAttenuation',55, ... % Magnitude constraints

'PassbandRipple',2, ...

'DesignMethod','kaiserwin', ... % Design method

'ScalePassband',false, ... % Design method options

'SampleRate',measure\_freq\_ecg2);

ecg2\_HP = filtfilt(HP\_filter, ecg2);

ecg2\_HP\_timecolumn = [ones(ecg2\_rows, 1), ecg2\_HP];

%verander enen naar tijdswaarden en plot

ecg2\_HP\_timecolumn = convert\_to\_time\_and\_plot(ecg2\_HP\_timecolumn, measure\_freq\_ecg2, ecg2\_rows);

title('Removing baseline wander from ecg2.mat')

% removing high frequency noise:

LP\_filter = designfilt('lowpassfir', ... % Response type

'PassbandFrequency',40, ... % Frequency constraints

'StopbandFrequency',40.2, ...

'StopbandAttenuation',55, ... % Magnitude constraints

'PassbandRipple',2, ...

'DesignMethod','kaiserwin', ... % Design method

'ScalePassband',false, ...

'SampleRate',measure\_freq\_ecg2);

ecg2\_LP = filtfilt(LP\_filter, ecg2);

ecg2\_LP\_timecolumn = [ones(ecg2\_rows, 1), ecg2\_LP];

%verander enen naar tijdswaarden en plot

ecg2\_LP\_timecolumn = convert\_to\_time\_and\_plot(ecg2\_LP\_timecolumn, measure\_freq\_ecg2, ecg2\_rows);

title('Removing high frequency noise from ecg2.mat')

% removing both at the same time:

BP\_filter = designfilt('bandpassfir', ... % Response type

'StopbandFrequency1',0.3, ... % Frequency constraints

'PassbandFrequency1',0.5, ...

'PassbandFrequency2',40, ...

'StopbandFrequency2',40.2, ...

'StopbandAttenuation1',55, ... % Magnitude constraints

'PassbandRipple',2, ...

'StopbandAttenuation2',55, ... % Magnitude constraints

'DesignMethod','kaiserwin', ... % Design method

'ScalePassband',false, ...

'SampleRate',measure\_freq\_ecg2);

ecg2\_BP = filtfilt(BP\_filter, ecg2);

ecg2\_with\_bp = [ones(ecg2\_rows, 1), ecg2\_BP];

%verander enen naar tijdswaarden en plot

ecg2\_with\_bp = convert\_to\_time\_and\_plot(ecg2\_with\_bp, measure\_freq\_ecg2, ecg2\_rows);

title('Removing baseline wander and high frequency noise from ecg2.mat')

%% Task 7

LP\_filter =designfilt('lowpassiir', ... % Response type

'PassbandFrequency',55, ... % Frequency constraints

'StopbandFrequency',55.2, ...

'PassbandRipple',2, ... % Magnitude constraints

'StopbandAttenuation',55, ...

'DesignMethod','ellip', ... % Design method

'MatchExactly','both', ... % Design method options

'SampleRate',measure\_freq\_ecg);

ecg\_LP = filtfilt(LP\_filter, ecg);

ecg\_LP\_t = [ones(length(ecg\_LP), 1), ecg\_LP];

%verander enen naar tijdswaarden en plot

ecg\_LP\_t = convert\_to\_time\_and\_plot(ecg\_LP\_t, measure\_freq\_ecg, length(ecg\_LP));

title('Removing high frequency noise from ecg.mat')

ecg\_downsampled = downsample(ecg\_LP, 10);

ecg\_downsampled\_timecolumn = [ones(length(ecg\_downsampled), 1), ecg\_downsampled];

%verander enen naar tijdswaarden en plot

ecg\_downsampled\_timecolumn = convert\_to\_time\_and\_plot(ecg\_downsampled\_timecolumn, 6, length(ecg\_downsampled));

title('Downsampling cleaned ecg.mat bij factor 10')

%% personal functions:

function output = convert\_to\_time\_and\_plot(input, freq, rows)

output = input;

for i = 1: rows

output(i,1) = i \* 1/freq;

end

figure

plot(output(:,1), output(:,2));

title("ECG data infv de tijd in seconden");

xlabel("Tijd(s)");

ylabel("ECG amplitude data");

end

function [amp, freq] = calculate\_FFT(input, rijen, mfreq)

FFT = fft(input);

P2 = abs(FFT / rijen);

P1 = P2(1:rijen/2+1)/length(rijen);

P1(2:end-1) = 2 \* P1(2:end-1);

f = mfreq \*(0:(rijen/2))/rijen;

figure

hold on

plot(f, P1)

[amp, freq] = findpeaks(P1, f, 'MinPeakDistance',9 , 'MinPeakHeight', 0.02);

plot(freq, amp, 'x');

xlabel('Frequency [Hz]');

ylabel('Amplitude');

legend('FFT');

hold off

end

function [Notch, z1, z2] = calculate\_notch\_with\_conj(input, degrees, a, freq, rows)

radials = degrees \*2\*pi / freq;

z1 = cos(radials) + 1j \* sin(radials);

z2 = conj(z1);

Notch = [input(:,1), zeros(rows,1)];

Notch(1,2) = input(1,2);

Notch(2,2) = input(2,2) - input(1,2)\*(z1+z2) + Notch(1,2) \*(a\*z1+a\*z2);

Notch(3:end, 2) = input(3:end,2) - input(2:end-1,2)\*(z1+z2) + Notch(2:end-1,2) \*(a\*z1+a\*z2)+ ( input(1:end-2,2) - Notch(1:end-2,2) \* a^2 )\* z1\*z2;

end

function [Notch\_ecg, z1, z2] = clean\_notch(thresh, ecg\_freq, Notch\_ecg, a, freq, rows)

freq\_degrees = ecg\_freq(find(ecg\_freq > thresh,1));

while( freq/2 > thresh)

[Notch\_ecg, z1,z2] = calculate\_notch\_with\_conj\_clean(Notch\_ecg, freq\_degrees, a, freq, rows);

[~,freq\_ecg] = calculate\_FFT(Notch\_ecg(:,2), rows, freq);

title('filtered more than once with the Notch filter')

freq\_degrees = freq\_ecg(find(freq\_ecg > thresh,1));

thresh = freq\_degrees - 50;

[~,n] = size( freq\_degrees);

if n == 0

thresh = freq;

end

end

end

function [Notch, z1, z2] = calculate\_notch\_with\_conj\_clean(input, degrees, a, freq, rows)

radials = degrees \*2\*pi / freq;

z1 = cos(radials) + 1j \* sin(radials);

z2 = conj(z1);

Notch = [input(:,1), zeros(rows,1)];

Notch(1,2) = input(1,2);

Notch(2,2) = input(2,2) - input(1,2)\*(z1+z2) + Notch(1,2) \*(a\*z1+a\*z2);

Notch(3:end, 2) = input(3:end,2) - input(2:end-1,2)\*(z1+z2) + Notch(2:end-1,2) \*(a\*z1+a\*z2)+ ( input(1:end-2,2) - Notch(1:end-2,2) \* a^2 )\* z1\*z2;

end

function impulse\_response(z1 ,z2)

Num = [1 -z1-z2 z1\*z2];

Denum = [1 -0.9\*z1-0.9\*z2 0.81\*z1\*z2];

figure

impz(Num,Denum,30);

title('impulse response');

figure

freqz(Num,Denum);

title('frequency response');

end