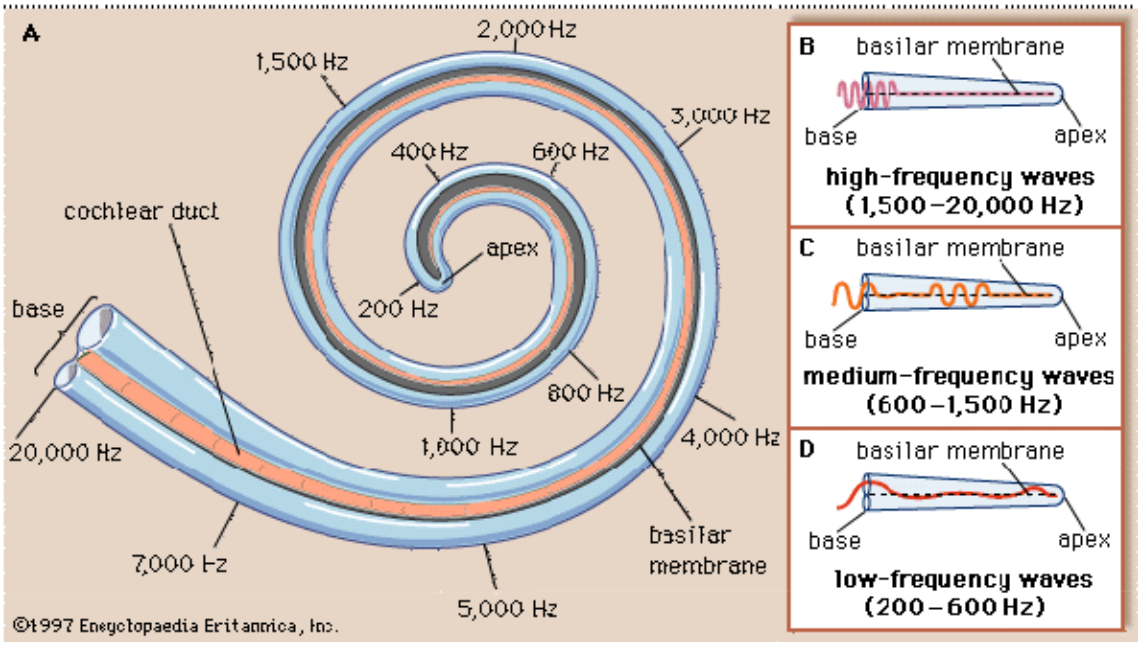
Blatt 6: CI Signalverarbeitung

Nicoletti Michele

Aufgabe 1

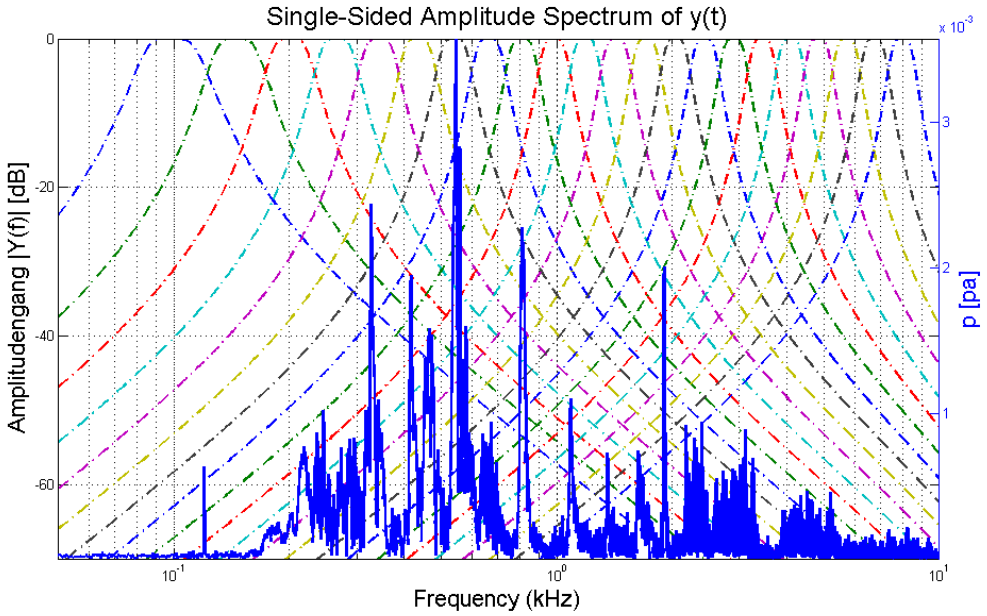
Für ein Cochlea Implantat soll die Tonotopie der menschlichen Cochlea signaltechnisch nachgebildet werden:



Implementieren sie eine Filterbank mit 3, 6, 12 und 22 Ausgängen, zeigen sie wie sich die eingelesenen Signale durch die unterschiedliche Anzahl an Kanälen ändern.
Da es sich um eine Vorverarbeitungsstufe für ein Cochleaimplantat handelt sind die Elektroden gleichmäßig über die Cochlea verteilt. Für die Frequenz zu Ort Zuweisung verwenden wir die Greenwood Funktion, die Breite der Filter wird anhand der ERB (Equivalent Rectangular Bandwidth) ermittelt (Siehe Appendix).

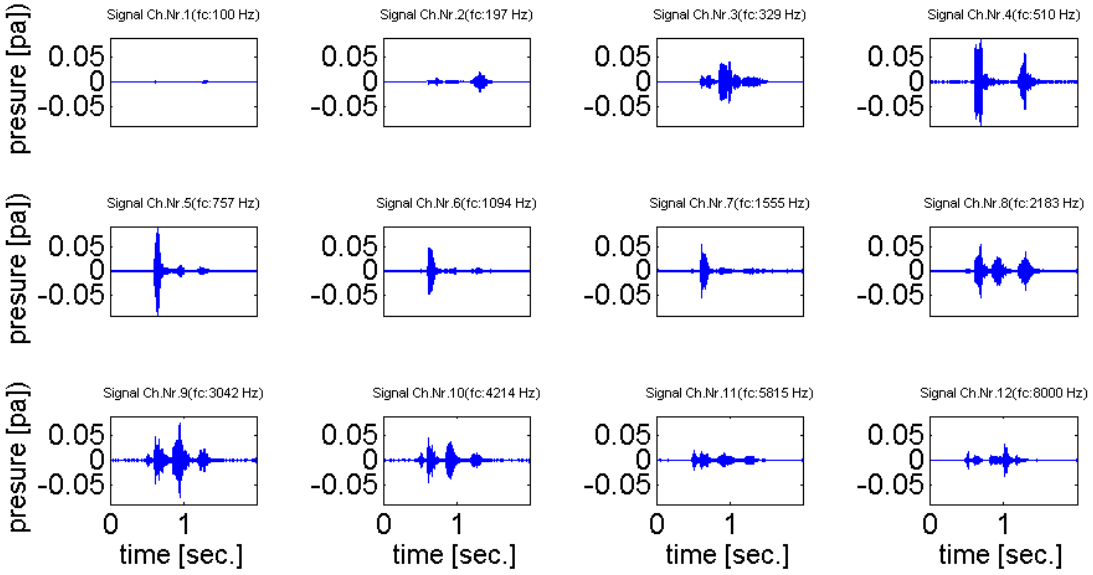
a) Implementieren Sie die Filterbank mit Butterworth IIR Filtern und Ploten sie die Frequenzgänge der Filter für 3 und 22 Kanäle.

Lösung: 22 Kanal Filterbank



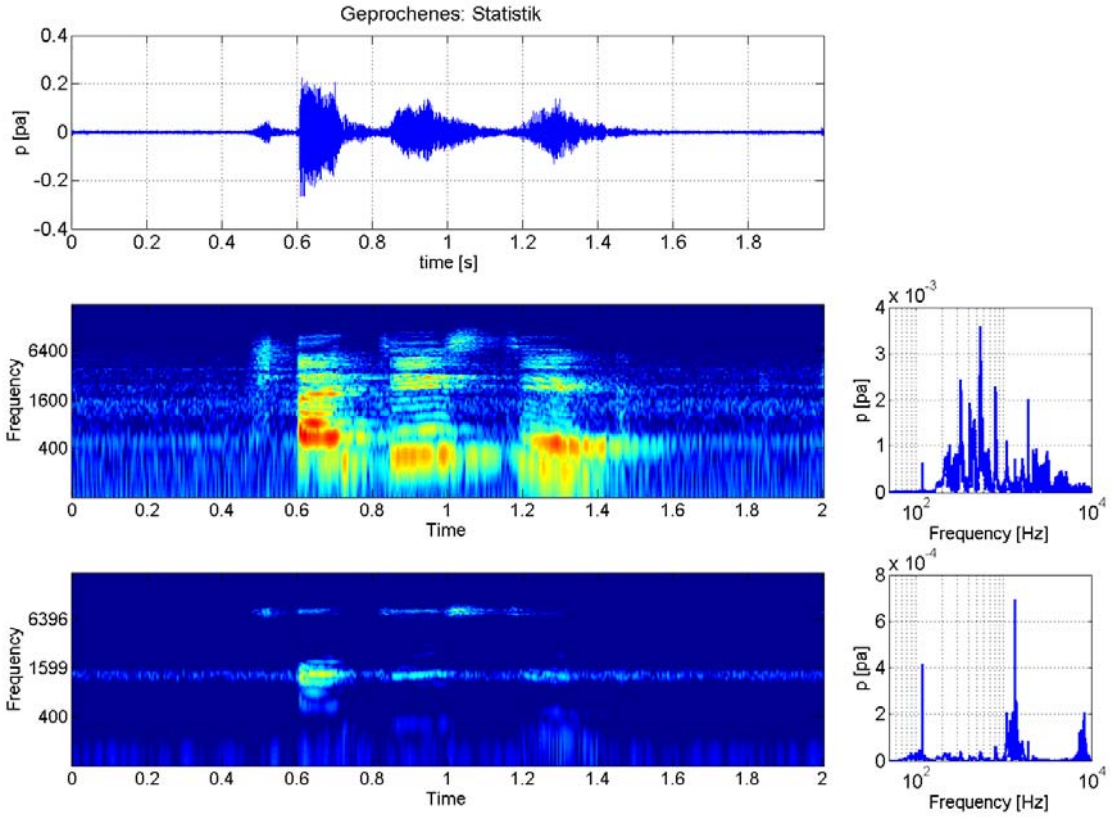
b) Nehmen Sie ein beliebiges Wort mit ihrer Soundkarte auf und verarbeiten Sie es mit ihrer Filterbank. Ploten Sie die Ausgänge der jeweiligen Kanäle für eine 6 und eine 12 Kanal Filterbank

Lösung: Gesprochenes ‚Statistik' mit 12 Kanal Filterung



c) Fügen sie die Kanalausgänge nach der Filterung wieder zusammen. Ploten sie das Langzeitspektren und das Spectrogram (Kurzzeitspektren) eines beliebigen aufgenommenen Wortes, vor und nach der Filterung, mit einer 3 und einer 12 Kanal Filterbank.

Lösung: Gesprochenes ‚Statistik' mit 3 Kanal Filterung



Appendix:

Signal einlesen:

```
%% Einlesen eins Audio Signal über den Mikrofon Eingang der Sound-Karte ###
      AufnahmeZeit=2; % [s]         Länge der aufgenommenen Sequenz
      fs=48000;       % [Hz]        Sampling Frequenz der Aufnahme
      bit=16;         % [bit]       Auflösung in bit

      % Generierung eins Audioobjektes
      recObj = audiorecorder(fs, bit, 1);
      get(recObj)

      % Starten der Audioaufnahme
      disp('Start speaking.')
      pause(2);                     % zwei Sekunden Pause zum Luft holen :-)
      recordblocking(recObj, AufnahmeZeit);
      disp('End of Recording.');

      % Abspielen der Audio Aufnahme
      play(recObj);

      % Abspeichern der Daten in ein double-precision array.
      myRecording = getaudiodata(recObj,'double');
```

Filter Bank:

```
% Greenwood Parameters (1990)
      EarQ = 7.23824;
      minBW = 22.8509;
      order = 1;
% Berechnung der ERB & Mittenfrequenzen
      % numChannels: Anzahl der Kanäle
      % loFreq: Niedrigste Frequenz (zb. 100 Hz)-> Erster Filter
      % hiFreq: Höchste Frequenz (zb. 8000 Hz)  -> Letzter Filter
      ERBlo = ((loFreq/EarQ)^order + minBW^order) ^ (1/order);
      ERBhi = ((hiFreq/EarQ)^order + minBW^order) ^ (1/order);
      overlap = (ERBhi/ERBlo)^(1/(numChannels-1));

% Equivalent Rectangular Bandwidth
      ERB = ERBlo * (overlap.^(0:numChannels-1));

% Mittenfrequenzen
      cf = EarQ*(((ERB.^order) - (minBW.^order)).^(1/order));
            .
            .
            .
% Erzeugen der Bandpassfilter (als IIR)
      ftype = 'bandpass';
      f_Ny=fs/2;              % Nyquist Frequenz
            .
            .
 for i=1:numChannels
      % Filter Ordnung 2
      % N_l_Cf : untere Grenzfrequenz (!!Grenzfrequenz sind
      % normiert auf die Nyquist Frequenz)
      [b,a]=butter(2,[N_l_Cf(i) N_u_Cf(i)],ftype);
```

```
      % Signal Filtern
      Filterausgang(:,i)=filter(b,a,x);

      % Frequenzgang und Gruppenlaufzeit
      [H(:,i),w]        = freqz     (b, a, n_ff, fs);
      [Gr(:,i), w_gd]   = grpdelay  (b, a, n_ff, fs);
end
% Achtung !!!!!! Bei Filtern Höherer Ordnung (>2)
% Transfer Function Design kann instabile Filter verursachen
% (Numerische Limitation in Matlab)in diesem Fall ist ein
% Zero-Pole-Gain Design zu bevorzugen
```

Plots:

```
% Ploten des Amplitudenspektrums
            % FFT des Signals
            n = floor(length(x)/2);
            %FFT (Normiert [0 1] bzg. Anzahl der Abtastpunkte)
            FFTtmp = fft(x');
            FFT = abs(FFTtmp(:,1:n+1)./n);% Real Anteil und Skalierung (Hz)
            f=(0:n)/n*fs/2;
      semilogx(f, FFT)        % mit logarithmierter Frequenz Achse
            % Darzustellender Bereich (f_min f_max   FFT_min FFT_max)
            axis([50 10000 min(FFT) max(FFT)]);
            xlabel('\fontsize{16}Frequency (Hz)');
            ylabel('\fontsize{16}\fontsize{16}p^{2} [pa^{2}]')
            grid on

% Ploten des Frequenzgang in KHz und dB
      plot((w/1e3),20*log10(abs(H)),'LineWidth',2.0)

% Ploten des Spectrogram
            % fs: Abtastfrequenz
            % Länge des Hamming Fensters in Abtastpunkten (hier 10 ms)
            N_Fenster=10e-3*fs;
            % Länge der Fensterüberlappung in Abtastpunkten (hier 5 ms)
            Ueberlapp=5e-3*fs;
            % Länge der Ausgabewerte B und f
            n_out=1024;

      % Berechnung des Spectrogram von ‚signal'
      [B,f,t]=specgram(signal, n_ou, fs, N_Fenster, Ueberlapp);

      % Darstellung des Spectrogram in dB im Bereich zwischen dem
            % Maximalwert und -60 dB dessen.
            bmin= max(max(abs(B))) *10^-(60/20); % -60 dB des Max. Wertes
      imagesc(t, f, 20*log10( max(abs(B), bmin)/bmin ) );
            % Alternativ auch mit:
            pcolor (t, f, 20*log10 (P)

      % Da die Funktion specgram nur ein lineares Spektrum darstellt ist
      % zur Darstellung von Sprachsignale die folgende die Funktion besser
      % geeignet (zu downloaden unter Lehre):
      logfsgram(signal, n_out,fs, N_Fenster, Uberlap, loFreq,1000);
            caxis([-30 30])
```