R. Brigola, TH Nürnberg Georg Simon Ohm, 2014

Mathematica - Notebooks als Bonusmaterial zum Lehrbuch

[1] Rolf Brigola Fourier-Analysis und Distributionen,
Eine Einführung mit Anwendungen,
edition swk, Hamburg 2013

Beispiel zum Entwurf eines diskreten Notch-Filters nach der Methode der Bilinearen Transformation mit *Mathematica* und Test mit einem Audio-Beispiel

Referenzen zu Kapiteln, Abschnitten, Seitenzahlen beziehen sich im Folgenden meist auf das genannte Lehrbuch des Autors. Einige wenige weitere Referenzen findet man am Ende des Notebooks.

Die URL aller meiner *Mathematica*-Notebooks zur Fourier-Analysis ist:
http://www.stiftung-swk.de/mathematica/

In diesem Notebook wird eine serielle Verknüpfung von 3 Bandsperren-Filtern (Notch-Filtern) berechnet für die “Notch-Frequenzen” 233 Hz, 466 Hz und 699 Hz. Dies sind die (grob gemessenen) Frequenzen von Grundton und den ersten 2 Obertönen (Oktave und Quint oktaviert) von Vuvuzelas, wie sie immer bei TV-Übertragungen von Spielen der Fußball-WM 2010 aus Südafrika als ständiges Hintergrunddröhnen zu hören waren.

Wir testen das enstehende diskrete Filter 6 Ordnung, das wir mittels bilinearer Transformation aus den Analogfrequenzgängen von Notch-Filtern 2. Ordnung berechnen, mit einem Ausschnitt einer solchen Übertragung.

Filterberechnung mit der bilinearen Transformation

Ausgehend vom Frequenzgang eines Notch-Filters 2. Ordnung berechnen wir nach der im vorangehenden Notebook vorgestellten Methode die Koeffizienten der rationalen Übertragungsfunktion $H(z)= \sum_{m=0}^{N} a_m z^{-m} / \sum_{m=0}^{N} b_m z^{-m}$ des resultierenden diskreten Filters der Ordnung N=6. Man kann dann allein mit den Koeffizienten a_k , b_k unmittelbar eine Realisierung mit der zugehörigen Differenzengleichung nach Normierung des Koeffizienten b_0 auf b_0= 1 (vgl. ggf. [1],S. 304) programmieren:

(4) $$y_n = a_0 x_n + \sum_{m=1}^{N} (a_m x_{n-m} - b_m y_{n-m})$$

für Input-Samples x_n und Output-Samples y_n und verschwindende Anfangsbedingungen. Statt dies zu tun, verwenden wir die dafür bereits vorliegende ***Mathematica*-Routine RecurrenceFilter**.

Filterberechnung mit bilinearer Transformation

Wir geben den Frequenzgang R eines Notch-Filters 2. Ordnung vor, d.h.

$$R[\omega\_] := \left(1 + (i\,\omega/\text{wg})^2\right) \Big/ \left(1 + i\,\omega \Big/ (\text{wg}\ Q) + (i\,\omega/\text{wg})^2\right)$$

für die Notch-Kreisfrequenz wg.Damit berechnen wir die Koeffizienten von Zähler- und Nenner-Polynom des diskreten Filters, das durch bilineare Transformation entsteht. Die Vorverzerrung wird so gewählt, dass $R(wg) = H(e^{i\,wg\,A})$ für die Übertragungsfunktion H des diskreten Filters gilt. Als "Qualitätsfaktor" Q wählen wir Q=5. Ansonsten wählen wir K=1, wg steht vorläufig als Parameter für die Notch-Frequenzen, A=1/44100 für die vorliegende .wav-Testaufnahme. Wir setzen im Folgenden zur Übersichtlichkeit $L = \cot(\omega_g A/2)$.

Nun zur Berechnung:

Um eine möglichst übersichtliche Darstellung mit *Mathematica* zu erreichen, substituiere ich z^{-1} durch w und verwende den ***Mathematica*-Befehl Cancel**: Das Ergebnis ist dann eine in w rationale Funktion, aus der wir mit *Mathematica* leicht die benötigten Koeffizienten im Zähler und im Nenner als Listen extrahieren können.

a) Das diskrete Notchfilter mittels bilinearer Transformation,
zunächst noch ohne explizite Vorgabe der Notch-Frequenzen wg1,.., wg3 und damit von L. Der Frequenzgang R hat bei wg eine Nullstelle.
Wichtig: *Da der Frequenzgang einer digitalen Audio-Aufnahme mit der Abtastfrequenz f Hz als Funktion der Kreisfrequenz 2π f -periodisch ist, müssen die Bandbreite der Aufnahme und die des Filters zusammenpassen. Überlegen Sie, welche Effekte unterschiedliche Perioden haben können. Ich verwende nachfolgend ein File im .wav-Format mit der CD-Abtastfrequenz f=44100 Hz, daher ist A=1/44100 die Zeitdauer in s zwischen zwei Abtastwerten. Für das Filter wähle ich das gleiche Abtastintervall A.*

In[1]:=
```
Q := 5; wg1 := 466 π; wg2 := 932 π; wg3 := 1398 π; A := 1 / 44100;
L := Cot[wg A / 2];
R[ω_] := (1 + (i ω / wg)^2) / (1 + i ω / (wg Q) + (i ω / wg)^2);
(* bei der Notch-Frequenz wg hat R eine Nullstelle *)
B[w_] := wg L (1 - w)/(1 + w); (* bilineare Transformation, w=z^-1,
L=cot(wg A/2); 2V/A mit V=ωgA/2cot[wg A/2] gekürzt *)
HH[w_] = Cancel[R[B[w] / i] ]
(* in w rationale Übertragungsfunktion des diskreten Filters *)
zaehlerkoeff = N[CoefficientList[Numerator[%], w]]
(* Unnormierte Koeffizientenliste des Zählerpolynoms *)
nennerkoeff = N[CoefficientList[Denominator[%%], w] ]
 (* Unnormierte Koeffizientenliste des Nennerpolynom *)
```

Out[5]=
$$\left(5\left(1+2w+w^2+\text{Cot}\left[\frac{wg}{88\,200}\right]^2-2w\,\text{Cot}\left[\frac{wg}{88\,200}\right]^2+w^2\,\text{Cot}\left[\frac{wg}{88\,200}\right]^2\right)\right)\Big/$$
$$\left(5+10w+5w^2+\text{Cot}\left[\frac{wg}{88\,200}\right]-w^2\,\text{Cot}\left[\frac{wg}{88\,200}\right]+\right.$$
$$\left.5\,\text{Cot}\left[\frac{wg}{88\,200}\right]^2-10w\,\text{Cot}\left[\frac{wg}{88\,200}\right]^2+5w^2\,\text{Cot}\left[\frac{wg}{88\,200}\right]^2\right)$$

Out[6]=
```
{5. + 5. Cot[0.0000113379 wg]^2,
 10. - 10. Cot[0.0000113379 wg]^2, 5. + 5. Cot[0.0000113379 wg]^2}
```

Out[7]=
```
{5. + Cot[0.0000113379 wg] + 5. Cot[0.0000113379 wg]^2,
 10. - 10. Cot[0.0000113379 wg]^2,
 5. - 1. Cot[0.0000113379 wg] + 5. Cot[0.0000113379 wg]^2}
```

b) Nun sukzessive die einzelnen Notch-Frequenzen eingesetzt und die jeweiligen Koeffizienten der “Teilfilter 2. Ordnung” berechnet und in a1,..,a3, b1,...,b3 zwischengespeichert

In[8]:=
```
wg = wg1
a1 = N[zaehlerkoeff]
b1 = N[nennerkoeff]
R1[w_] = R[w];
```

Out[8]= $466\,\pi$

Out[9]= $\{18\,150.,\ -36\,279.9,\ 18\,150.\}$

Out[10]= $\{18\,210.2,\ -36\,279.9,\ 18\,089.7\}$

In[12]:=
```
wg = wg2
a2 = N[zaehlerkoeff]
b2 = N[nennerkoeff]
R2[w_] = R[w];
```

Out[12]= $932\,\pi$

Out[13]= $\{4538.74, -9057.48, 4538.74\}$

Out[14]= $\{4568.85, -9057.48, 4508.63\}$

In[16]:=
```
wg = wg3
a3 = N[zaehlerkoeff]
b3 = N[nennerkoeff]
R3[w_] = R[w];
```

Out[16]= $1398\,\pi$

Out[17]= $\{2018.14, -4016.29, 2018.14\}$

Out[18]= $\{2038.21, -4016.29, 1998.08\}$

c) Nun die serielle Verknüpfung der 3 Filter:
Wir berechnen die Zähler- und Nenner-Koeffizienten der resultierenden Übertragungsfunktion H und speichern die (umnormierten) Koeffizienten unter a für die Zählerkoeffizienten und b für die des Nenners.

In[20]:=
```
p1[w_] := Sum[a1[[k]] w^(k-1), {k, 1, 3}]
p2[w_] := Sum[a2[[k]] w^(k-1), {k, 1, 3}]
p3[w_] := Sum[a3[[k]] w^(k-1), {k, 1, 3}]
q1[w_] := Sum[b1[[k]] w^(k-1), {k, 1, 3}]
q2[w_] := Sum[b2[[k]] w^(k-1), {k, 1, 3}]
q3[w_] := Sum[b3[[k]] w^(k-1), {k, 1, 3}]

H[w_] = Collect[(p1[w] p2[w] p3[w]), w] / Collect[(q1[w] q2[w] q3[w]), w]
zaehlerkoeff = N[CoefficientList[Numerator[H[w]], w]]
(* Unnormierte Koeffizientenliste des Zählerpolynoms *)
nennerkoeff = N[CoefficientList[Denominator[H[w]], w] ]
 (* Unnormierte Koeffizientenliste des Nennerpolynom *)
```

Out[26]= $\left(1.6625\times10^{11} - 9.94939\times10^{11}\,w + 2.48351\times10^{12}\,w^2 - 3.30965\times10^{12}\,w^3 + 2.48351\times10^{12}\,w^4 - 9.94939\times10^{11}\,w^5 + 1.6625\times10^{11}\,w^6\right) / \left(1.69578\times10^{11} - 1.00818\times10^{12}\,w + 2.49997\times10^{12}\,w^2 - 3.30957\times10^{12}\,w^3 + 2.46701\times10^{12}\,w^4 - 9.81777\times10^{11}\,w^5 + 1.62963\times10^{11}\,w^6\right)$

Out[27]= $\{1.6625\times10^{11}, -9.94939\times10^{11}, 2.48351\times10^{12}, -3.30965\times10^{12}, 2.48351\times10^{12}, -9.94939\times10^{11}, 1.6625\times10^{11}\}$

Out[28]= $\{1.69578\times10^{11}, -1.00818\times10^{12}, 2.49997\times10^{12}, -3.30957\times10^{12}, 2.46701\times10^{12}, -9.81777\times10^{11}, 1.62963\times10^{11}\}$

In[29]:=
```
a = N[zaehlerkoeff / nennerkoeff[[1]]]
(* Die Koeffizienten des Zählers von H *)
b = N[nennerkoeff / nennerkoeff[[1]]]  (* Die Koeffizienten des Nenners von H *)
```

Out[29]= {0.980375, −5.86714, 14.6452, −19.5169, 14.6452, −5.86714, 0.980375}

Out[30]= {1., −5.94522, 14.7423, −19.5165, 14.5479, −5.78952, 0.960988}

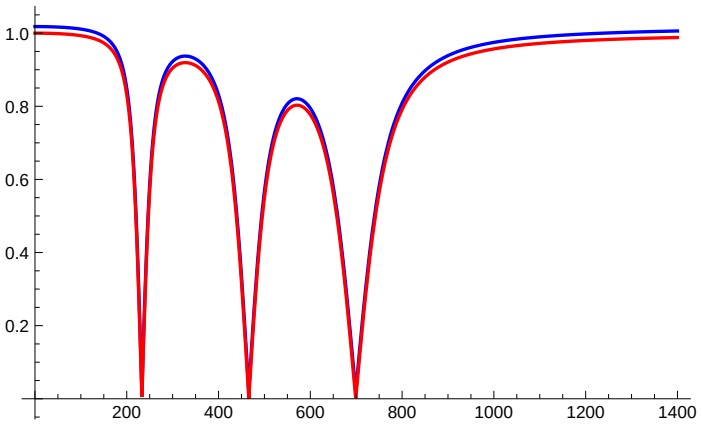
In[31]:=
```
R4[w_] = Simplify[R1[w] R2[w] R3[w]]
 (* Frequenzgang des Analogfilters mit den 3 gesperrten Frequenzen *)
```

Out[31]= $\left(125 \left(217\,156\,\pi^2 - w^2\right) \left(868\,624\,\pi^2 - w^2\right) \left(1\,954\,404\,\pi^2 - w^2\right)\right) / \left(\left(1\,085\,780\,\pi^2 + 466\,i\,\pi\,w - 5\,w^2\right) \left(4\,343\,120\,\pi^2 + 932\,i\,\pi\,w - 5\,w^2\right) \left(9\,772\,020\,\pi^2 + 1398\,i\,\pi\,w - 5\,w^2\right)\right)$

d) Hier der Amplitudengang des berechneten Filters in Rot und der des Analogfilters, von dem ausgegangen wurde in Blau, geplottet mit einem kleinen Offset zur besseren Unterscheidung.
Die Frequenzen, bei denen das Filter zu “Auslöschung” im gefilterten Signal führt, sind deutlich zu sehen.

In[32]:=
```
plot3a = Plot[Abs[R4[ 2 π f ]] + 0.018, {f, 0, 1400},
   PlotRange → All, PlotStyle → Directive[Blue, Thickness[0.005]] ];
plot3 = Plot[Abs[H[Exp[-I 2 π f A]]], {f, 0, 1400}, PlotRange → All,
   PlotStyle → Directive[Red, Thickness[0.005]] ] ;
Show[{plot3a, plot3}](* Amplitudengang der Filters,
über Frequenzen f in Hz aufgetragen: Blau Analogfilter,
rot diskretes Filter *)
```

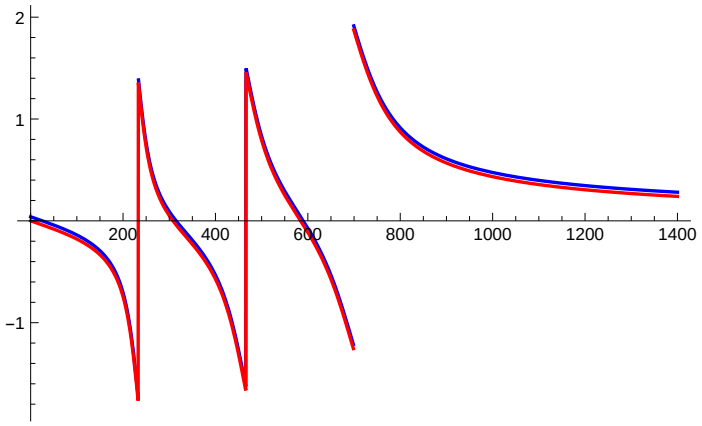
Out[34]=



In ähnlicher Weise arbeiten Mehrband-Audio-Equalizer, mit denen in gewissen Frequenzbändern verstärkt, in anderen gedämpft wird.
Unten noch die zugehörigen Phasengänge wieder in Blau mit etwas Offset und Rot beim diskreten Filter

In[35]:=
```
plot5 = Plot[Arg[R4[ 2 π f ]] + 0.04, {f, 0, 1400},
    PlotRange → All, PlotStyle → Directive[Blue, Thickness[0.005]] ];
plot6 = Plot[Arg[H[Exp[-I 2 π f A]]], {f, 0, 1400}, PlotRange → All,
    PlotStyle → Directive[Red, Thickness[0.005]] ] ;
Show[{plot5, plot6}]
```

Out[37]=



Test mit einer Audio-Aufnahme

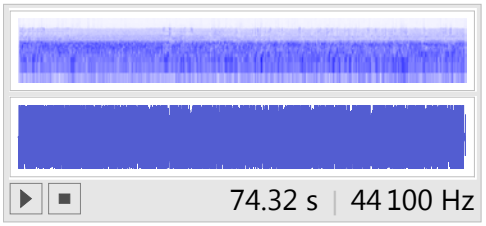
Wir laden das Audio-File WM_Vuvuzela.wav und hören hinein.
Der Vuvuzela-Lärm ist deutlich zu hören. Beim Download
müssen Sie nachfolgend den Pfad angeben, wohin Sie
das File lagern.
Nach dem Import können Sie das File in *Mathematica*
mit dem Sound-Befehl anhören.

In[38]:=
```
SetDirectory["D:/Mathematica"]
(* Beachten LW:/Verzeichnis, nicht LW:\Verzeichnis  *)
```

Out[38]=
```
D:\Mathematica
```

In[39]:=
```
data := Import["WM_Vuvuzela.wav"]
Sound[data]
```

Out[40]=



Nun die Filterung mit den Filterkoeffizienten wie oben berechnet mittels *Mathematica*-Befehl RecurrenceFilter.

Vergleichen Sie beide Aufnahmen vor und nach der Filterung.
Natürlich bekommt man “breitbandiges Rauschen” aus einer Aufnahme
kaum je mehr heraus, wenn es erstmal drin ist, die Vuvuzela-Frequenzen
sind aber für meine Ohren gut herausgefiltert.

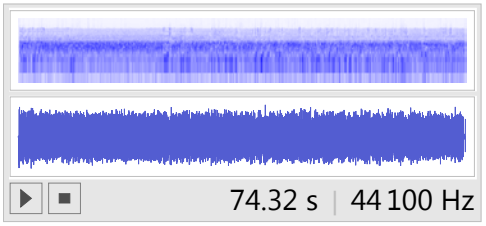
Wenn Sie möchten, vergleichen Sie das Ergebnis mit anderen Filtern,
die für diesen Zweck in 2010 vielfach ins Internet gestellt wurden.
(Testen Sie dabei ein Filter nicht nur an einer solchen Aufnahme.
Ich habe oft gefunden, dass solche Filter für das eine Testbeispiel, das
dazu als Demo geliefert wurde, ein durchaus brauchbares Ergebnis hatten,
aber schon bei einer zweiten Aufnahme sehr kläglich waren.)

Ich stelle Ihnen 2 Aufnahmen zum Test auf meine Website:
Die erste wie hier als *WM_Vuvuzela.wav*, eine zweite als *Vuvuzela.wav*

Nun zum Ergebnis:

In[41]:=
```
data2 = RecurrenceFilter[{b, a}, data]
Export["Soundgefiltert.wav", data2]
```

Out[41]=



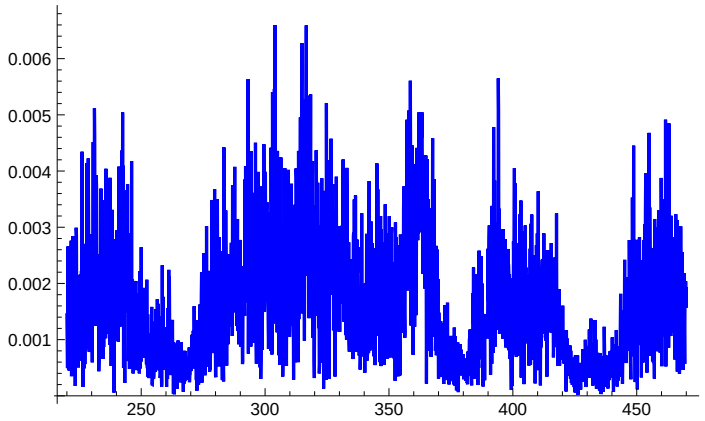
Out[42]= Soundgefiltert.wav

Nachfolgend noch ein Blick auf Ausschnitte aus den
Betragsspektren - jeweils im Bereich von 220 bis 470 Hz -
in Blau vom Ausgangssignal, in Rot vom gefilterten Signal.
Man erkennt deutlich die Amplitudendämpfungen um 233 Hz
und 466 Hz.

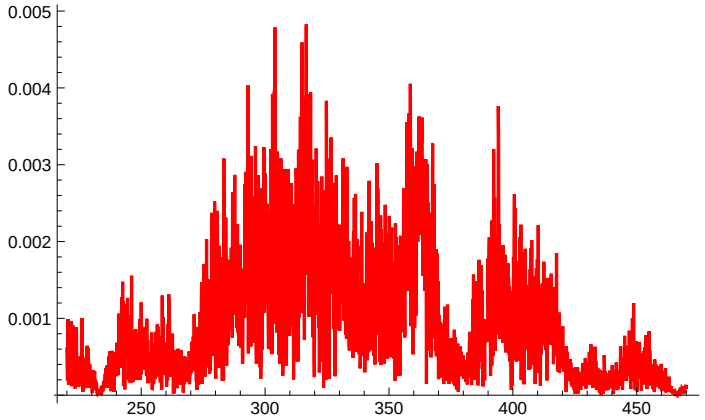
In[43]:=
```
dataraw = Import["WM_Vuvuzela.wav", "Data"];
(*data2raw=RecurrenceFilter[{b,a},dataraw];*)
data2raw = Import["Soundgefiltert.wav", "Data"];
differenz =
  Flatten[dataraw][[1 ;; 8 * 44 100]] - Flatten[data2raw][[1 ;; 8 * 44 100]];
ausschnittdata = Flatten[dataraw][[1 ;; 8 * 44 100]];
ausschnittgefiltert = Flatten[data2raw][[1 ;; 8 * 44 100]];
```

In[48]:=
```
specdata = Chop[Abs[Fourier[ausschnittdata, FourierParameters → {-1, -1}]]];
specfiltered =
  Chop[Abs[Fourier[ausschnittgefiltert, FourierParameters → {-1, -1}]]];
ListLinePlot[Flatten[Abs[specdata]][[1761 ;; 3761]], PlotRange → All,
 DataRange → {220, 470}, PlotStyle → Directive[Blue, Thickness[0.005]]]
ListLinePlot[Flatten[Abs[specfiltered]][[1761 ;; 3761]], PlotRange → All,
 DataRange → {220, 470}, PlotStyle → Directive[Red, Thickness[0.005]]]
```

Out[50]=



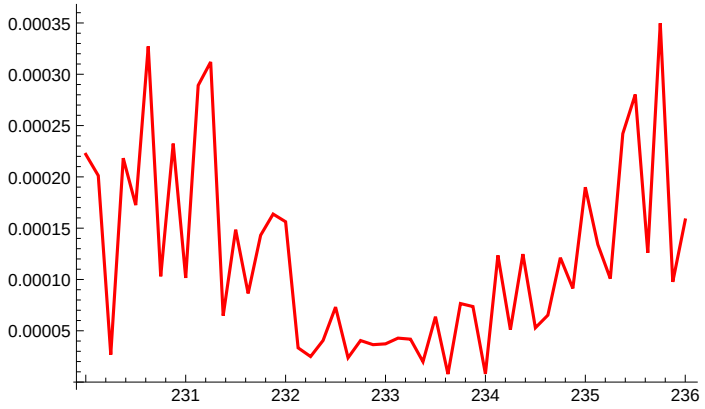
Out[51]=



Hier ein Ausschnitt des Betragsspektrums des gefilterten Signals, als ListLinePlot im Bereich von 230-236 Hz. Die Kerbe bei 233 Hz ist sichtbar. Durch DFT-Alias und Leakage-Effekte ist aber der Wert bei 233 Hz nicht exakt Null.

In[52]:=
```
ListLinePlot[Flatten[Abs[specfiltered]][[1841 ;; 1889]], PlotRange → All,
 DataRange → {230, 236}, PlotStyle → Directive[Red, Thickness[0.005]]]
```

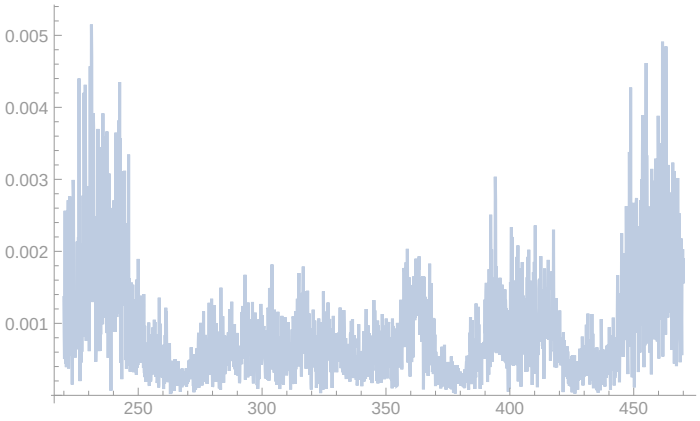
Out[52]=



Schließlich noch ein Ausschnitt der betragsmäßigen Differenz
der beiden obigen Spektren vom Signal und seiner gefilterten Version
im Bereich von 220 bis 470 Hz durch Näherung mit einer DFT über 8 s
mit 8*44100 DFT-Punkten.
Man erkennt die “Peaks” der Vuvuzela bei 233 Hz und 466 Hz in der Differenz
und die Differenz zeigt, dass durch Rundungsfehler bei der Filterung - sieht man
von Alias und Leakage-Effekten bei der DFT-Näherung einmal ab - und natürlich
durch die neue Quantisierung beim Speichern der gefilterten Version als .wav-File
ein auch hörbares **zusätzliches Rauschen in die Audio-Aufnahme** eingebracht wird.

```
absdiffspectrum = Chop[Abs[Fourier[differenz, FourierParameters → {-1, -1}]]];
ausschnittspectrum = Flatten[absdiffspectrum][[1761 ;; 3761]];
ListLinePlot[ausschnittspectrum, PlotRange → All, DataRange → {220, 470}]
```

Out[55]=



Literatur: Neben [1] empfehle ich zur Vertiefung

[2] A.V. Oppenheim, R.W.Schafer — Zeitdiskrete Signalverarbeitung, Oldenbourg, 1999

[3] H.W. Schüßler — Digitale Signalverarbeitung 1, Springer, 2008

[4] H. Wupper — Einführung in die digitale Signalverarbeitung, Hüthig, 1998