Spektralanalyse eines Sounds

Buch:Höhere Mathematik sehen und verstehen, Haftendorn, Riebesehl, Dammer, Springer Spektrum, Feb. 2021

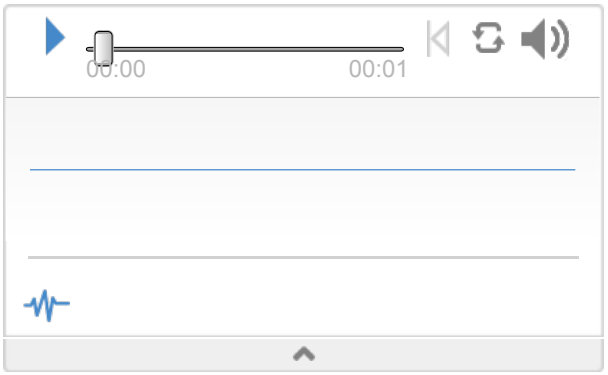
Datei Chordwav.nb zu Abschnitt 1.9.5 Seite 107, Abb. 1.70

Importieren des Sounds

Achtung: Bitte den Pfad für den Import so anpassen, dass das originale chord.wav geladen wird.

```
chord = Import["C:\\...\\chord.wav"]
```

Out[•]=



In[•]:=
```
chord1 = AudioData[chord]
```

Out[•]=
```
{{0., 0., 0.00006 1037, -0.00006 10352, 0.00006 1037, 0.00009 15555, 0.00012 2074, 0.00012 2
```
large output | show less | show more | show all | set size limit...

In[•]:=
```
rate = AudioSampleRate[chord][[1]]
```

Out[•]= 22 050

In[•]:=
```
amps = chord1[[1]]
```

Out[•]=
```
{0., 0., 0.00006 1037, -0.00006 10352, 0.00006 1037,
 0.00009 15555, 0.00012 2074, 0.00012 2074, 0.00006 1037, 0.00012 2074,
 0.00006 1037, ··· 24 209 ···, 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}
```
large output | show less | show more | show all | set size limit...

In[•]:=
```
Length[amps]
```

Out[•]= 24 231

In[ ]:=
```
Sound[SampledSoundList[amps, rate]]
```

Out[ ]=
```
Sound[SampledSoundList[{0., 0., 0.00006 1037, -0.00006 10352, 0.00006 1037,
    0.00009 15555, 0.00012 2074, 0.00012 2074, 0.00006 1037, 0.00012 2074,
    ··· 24 211 ···, 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}, 22 050 Hz]]
```

large output | show less | show more | show all | set size limit...

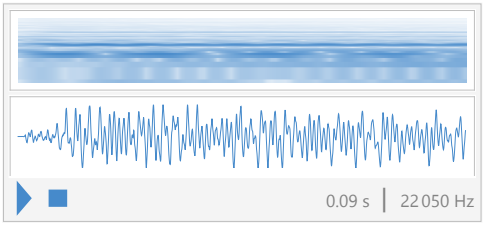
○ Analyse des Sounds mit diskreter Fourierzerlegung

In[ ]:=
```
n = 2048;
```

In[ ]:=
```
amps2 = Take[amps, n];
```

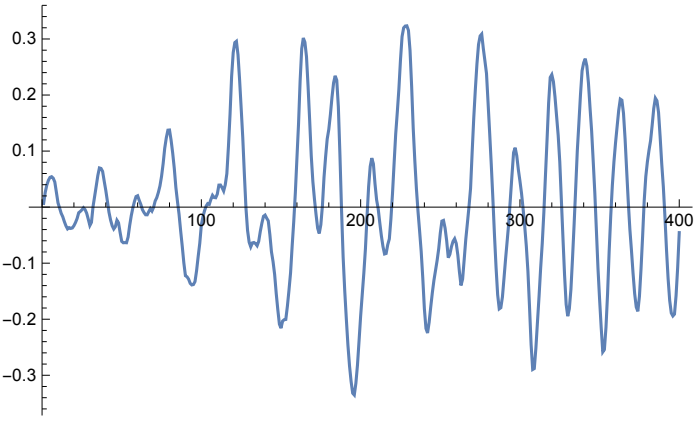
In[ ]:=
```
chordwavAkkord = Sound[SampledSoundList[amps2, rate]]
```

Out[ ]=



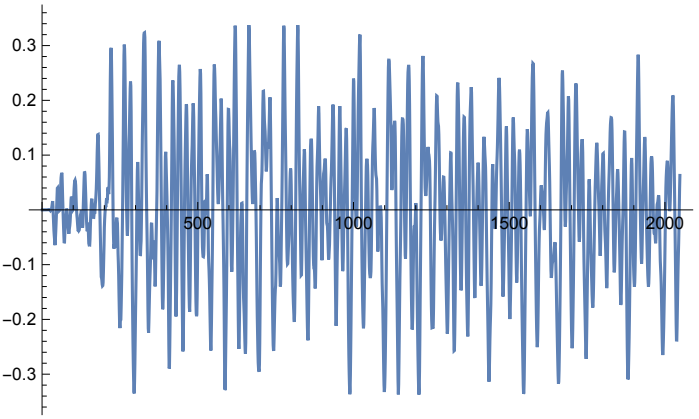
In[ ]:=
```
ListPlot[amps2[[Range[400] + 100]], Joined → True]
```

Out[ ]=



In[ ]:=
```
ListPlot[amps2, Joined → True]
```

Out[ ]=



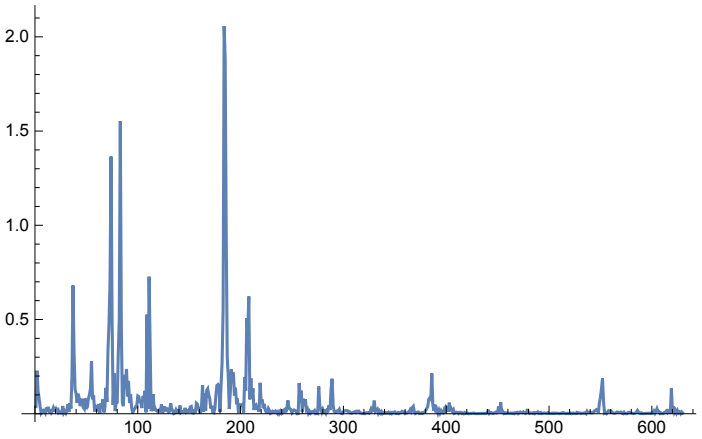
In[ ]:=
```
spec = FourierDCT[amps2];
```

In[•]:=
```
spes = FourierDST[amps2];
```

In[•]:=
```
Amp = √(spec² + spes²);
```

In[•]:=
```
ListPlot[Take[spec // Abs, 630] // Abs, Joined → True, PlotRange → All]
```

Out[•]=



○ Bestimmung der Peaks

In[•]:=
```
peaks = {37, 55, 74, 83, 109, 165, 184, 206, 219, 257, 276, 289, 386, 552, 619};
```

In[•]:=
```
{peaks, spec[[peaks - 1]], spec[[peaks]], spec[[peaks + 1]]} // Transpose // TableForm
```

Out[•]//TableForm=

37	−0.203074	0.681298	−0.330943
55	0.134197	−0.279094	0.076173
74	−0.700712	1.36492	0.508603
83	0.502899	−1.55323	−0.462512
109	0.00748677	−0.526299	0.025911
165	0.00969234	0.108288	−0.0151923
184	−0.554178	−2.05757	1.88328
206	0.122988	0.506533	0.442426
219	−0.0103407	0.164057	−0.0616186
257	−0.000626869	−0.162915	0.059615 7
276	−0.000585507	0.144471	0.042528 6
289	−0.132207	−0.185857	0.045934 4
386	−0.0769766	0.213823	0.045297 9
552	0.132909	−0.187717	−0.0637065
619	0.00139528	0.1341	0.011658 2

○ Frequenzen dazu

In[•]:=
```
(T = n / rate) // N
```

Out[•]=
```
0.0928798
```

In[•]:=
```
(ν = 1 / T) // N
```

Out[•]=
```
10.7666
```

○ Rekonstruktion des “chord”-Klanges aus den lautesten Frequenzen

In[•]:=
```
(freqs = ν peaks) // N
```
Out[•]=
```
{398.364, 592.163, 796.729, 893.628, 1173.56, 1776.49, 1981.05,
 2217.92, 2357.89, 2767.02, 2971.58, 3111.55, 4155.91, 5943.16, 6664.53}
```

geordnet nach Lautstärke

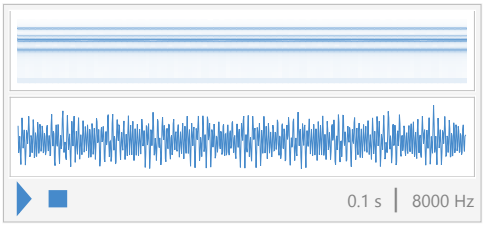
In[•]:=
```
peaksloud = #[[2]] & /@ ({spec[[peaks]] // Abs, peaks} // Transpose // Sort // Reverse)
```
Out[•]=
```
{184, 83, 74, 37, 109, 206, 55, 386, 552, 289, 219, 257, 276, 619, 165}
```

In[•]:=
```
freqsloud = (freqs = ν peaksloud) // N
```
Out[•]=
```
{1981.05, 893.628, 796.729, 398.364, 1173.56, 2217.92, 592.163,
 4155.91, 5943.16, 3111.55, 2357.89, 2767.02, 2971.58, 6664.53, 1776.49}
```

In[•]:=
```
fl = Take[freqsloud, 6]
```
Out[•]=
```
{1981.05, 893.628, 796.729, 398.364, 1173.56, 2217.92}
```

In[•]:=
```
chordfunc = Plus @@
  (Take[spec[[peaks]] // Sort // Reverse, 6] (Function[t, Cos[# 2 Pi t]][x] & /@ fl))
```
Out[•]=
```
0.213823 Cos[2503. x] + 0.506533 Cos[5005.99 x] + 0.681298 Cos[5614.83 x] +
 0.164057 Cos[7373.69 x] + 1.36492 Cos[12447.3 x] + 0.144471 Cos[13935.6 x]
```

In[•]:=
```
chordwavRebuild = Sound[Play[chordfunc, {x, 0, 0.1}]]
```
Out[•]=



○ Vielfache der Peaks testen

In[•]:=
```
set[peaks_, n_] := Line[#] & /@ ({{100 Range[Length[n peaks]], n peaks} // Transpose,
      {100 Range[Length[n peaks]] + 100, n peaks} // Transpose} // Transpose)
```

In[•]:=
```
Graphics[Table[set[peaks, n], {n, 1, 3}]]
```

Out[•]=