Nutzung digitaler Mikrofone als Signalquelle unter MATLAB®

Exemplarische Anwendung MATLAB® Tools zur Analyse von Schallsignalen

Using digital microphones as a signal source under MATLAB®

Exemplary application MATLAB® tools for the analysis of sound signals

Armin Rohnen
Hochschule München - Munich University of Applied Sciences
Faktultät 03 - Verbundlabor Fahrzeugtechnik, Akustik und Dynamik
Dachauer Straße 98b, 80335 München
eMail: rohnen@hm.edu

Inhaltsverzeichnis

1 Abstract

Im Bereich der Bühnen- und Studiotechnik finden digitale Mikrofone zunehmend Anwendung. Wobei die Bezeichnung ”digitales Mikrofon”hierbei irreführend ist. Das Mikrofon an sich unterscheidet sich im Aufbau und Funktionsweise nicht von bisherigen Technologien. Da die Analog-Digital-Umsetzung ebenfalls im Mikrofon erfolgt, ist dies namensgebend für diese Technologie. Standardtechnologie bei Messmikrofonen ist der analoge Mikrofonaufbau mit nachgeschaltetem Analog-Digital-Umsetzer.

Anders ist dies beim MV 240 digital von Microtech Gefell. Dies ist ein Messmikrofon-Vorverstärker mit integriertem Analog-Digital-Umsetzer. Hier steht direkt ein einkanalgiges USB-Ausgangssignal zur Verfügung. Die Analog-Digital-Umsetzung selbst erfolgt in zwei Pegelbereichen mit einer Wortbreite von jeweils 24 Bit, einmal für den oberen Pegelbereich und einmal für den unteren Pegelbereich. Durch den integrierten Prozessor werden die beiden Pegelbereiche zu einem einkanaligen digitalen USB-Ausgangssignal mit einer Wortbreite von 32 Bit zusammengefasst, welches bereits kalibrierten Schalldruckwerten entspricht. Ein Digit entspricht $1\mu Pa$ wodurch der gesamte Dynamikbereich von Messmikrofonkapseln abgedeckt wird, ohne dass eine Messbereichsumschaltung erforderlich wird. Es stehen die üblichen Abtastfrequenzen der Soundkarten zur Verfügung.

Für den Betrieb sind keine speziellen Treiber erforderlich und die Spannungsversorgung erfolgt direkt von der USB-Schnittstelle. Die für den Betrieb von Messmikrofonkapseln erforderliche Polarisationsspannung von 200 V wird intern erzeugt und kann bei Verwendung von Elektret-Messmikrofonkapseln abgeschaltet werden.

MATLAB® bietet mit der Data-Aquisition-Toolbox® einen einfachen Zugriff auf den Datenstrom angeschlossener Messgeräte, welche in diesem Beitrag dazu verwendet werden, Messaufgaben eines Schallpegelmessers [1, DIN EN 61672] und anderen Analysatoren für die Schallanalyse zu realisieren.

In the field of stage and studio technology, digital microphones are increasingly used. Whereby the term digital microphone is misleading. The microphone itself does not differ in structure and function from previous technologies. Since the analog-digital conversion also takes place in the microphone, this is the name given to this technology. Standard technology for measurement microphones is the analog microphone design with a downstream analog-to-digital converter.

This is different with the MV 240 digital from Microtech Gefell. This is a measurement microphone preamplifier with integrated analog-to-digital converter. Here, a single-channel USB output signal is directly available. The analog-to-digital conversion itself is done in two level ranges with a word width of 24 bit. One for the upper level range and one for the lower level range. The integrated processor converts the two level ranges to a single channel digital output signal with a word width of 24 bits. USB output signal with a word width of 32 bits, which corresponds to already calibrated sound pressure values. One digit corresponds to one micropascal, which covers the entire dynamic range of measuring microphone capsules without the need for range switching. There the usual sampling frequencies of the sound cards are available.

No special drivers are required for operation and power is supplied directly from the USB interface. The polarization voltage of 200 V required for the operation of measuring microphone capsules is generated internally and can be switched off when using electret microphone capsules.

MATLAB® offers with the Data Acquisition Toolbox® an easy access to the data stream of connected measurement devices, which are used in this paper to realize measurement tasks of a sound level meter [1, DIN EN 61672] and other analyzers for sound analysis.

2 Abkürzungen und Formelzeichen

$d(t)$	$[-]$	Digitalwert zum Zeitpunkt t
DFT	$[-]$	diskrete Fouriertransformation
f	$Hz-]$	Vektor der Frequenzachse
f_0	$[Hz]$	Teiloktav-Mittenfrequenz
f_1	$[Hz]$	Teiloktav untere Bandpassfilterfrequenz
f_2	$[Hz]$	Teiloktav obere Bandpassfilterfrequenz
f_{max}	$[Hz]$	maximale Analysefrequenz
f_s	$[Hz]$	Samplefrequenz, Abtastrate
FFT	$[-]$	Fast-Fouriertransformation
FM	$[-]$	Fenstermittelwert, Amplitudenkorrektur
$L(t)$	$[dB, dB(A)]$	Momentanschalldruckpegel
L_{eq}	$[dB, dB(A)]$	Equivalenter Schalldruckpegel
N	$[-]$	Blockgröße, Anzahl Abtastwerte
o	$[-]$	Overlap, Überlappung
s		Matrix der nichtnormierten, komplexen Fourierkoeffizienten
Sx		Matrix der Betragsspektren
t	$[s]$	Vektor der Zeitachse
T	$[-]$	Blocklänge, Mittelungszeit der Fouriertransformation
w	$[-]$	Fensterfunktion
$x(t)$		Messwert zum Zeitpunkt t
$\underline{X}(n)$		komplexe Fouriertransformierte

3 MATLAB® in Verbindung mit Messgeräten

MATLAB® bietet über verschiedene Toolboxen die Möglichkeit auf Datenströme kontinuierlich oder in Blöcken zugreifen zu können. Neben der Toolbox wird i. d. R. ein zusätzliches Hardware Support Paket benötigt.

Für die Nutzung des digitalen Mikrofons bietet sich an, dies über die Data-Aquisition-Toolbox® in Verbindung mit dem Hardware-Support für Windows-Soundkarten durchzuführen. Die Toolbox sowie das Hardware-Support-Paket müssen installiert sein, bevor das digitale Mikrofon an den PC angeschlossen wird.

Nach dem Start von MATLAB® lässt sich durch Eingabe von *daq.getDevices* im MATLAB® Command Window ein überblick über die verfügbaren (Mess-)Devices verschaffen. Es erfolgt eine Auflistung der verfügbaren Devices wie z. B. in Tabelle 1.

Tabelle 1: Liste der verfügbaren Devices für die Datenerfassung

index	Vendor	Device ID	Description
1	directsound	Audio0	DirectSound Primärer Soundaufnahmetreiber
2	directsound	Audio1	DirectSound Microphone Array (Realtek High Definition Audio(SST))
3	directsound	Audio2	DirectSound Mikrofon (32-Bit-Messmikrofon)
4	directsound	Audio3	DirectSound Primärer Soundtreiber
5	directsound	Audio4	DirectSound Lautsprecher (Realtek High Definition Audio(SST))

Hinter der Bezeichnung DirectSoundMikrofon$(32-\text{Bit}-\text{Messmikrofon})$ mit der Device-ID Audio2 verbirgt sich das digitale Messmikrofon MV240. Für die Nutzung als Messgerät muss dies lediglich mit wenigen Anweisungen konfiguriert werden.

```
messgeraet = daq('directsound');
addinput(messgeraet, 'Audio2', 1, 'Audio');
messgeraet.Rate = 48000;
```

Im weiteren erfolgt über die Anweisung

```
[data, time, ~] = read(messgeraet, seconds(20), 'OutputFormat', 'Matrix');
```

eine 20 Sekunden andauernde Messung.

Es liegen nun Messdaten in der Matrix data und zugehörige Zeitstempel in der Matrix time vor.

4 Skalierung der Audio-Messwerte

Numerisch stellt der Messwert eines Audio-Messgerätes im Wertebereich $\pm$ 1 den Aussteuerungsbereich des Messkanals dar. Um dies in einen physikalisch richtigen Messwert umzuwandeln bedarf es der Multiplikation mit dem Eingangsspannungsbereich und im Weiteren mit der Umrechnung von Spannungsmesswert über den Übertragungsfaktor (umgangssprachlich dem Kalibrierwert) auf den physikalischen Messwert. Der digitale Messmikrofonvorverstärker MV240 führt dies bereits intern im Vorverstärker durch. Das Ausgangssignal weist eine Auflösung von 1 μPa bei einer Wortbreite von $\pm 2^{31}$ auf und ist in der Signalverstärkung an die verwendete Mikrofonkapsel angepasst. Dies bedeutet im Umkehrschluss, dass beim Wechsel der Mikrofonkapsel der digitale Messmikrofonvorverstärker neu abgeglichen werden muss. Der Messwert $x(t)$ in Pascal berechnet sich zu

$$x(t) \;=\; d(t) \cdot \frac{2^{31}}{1000000} \tag{1}$$

Nach Umrechnung der Messwerte in Schallpegel, liefert das Diagramm "Schalldruck über Zeit"einen ersten visuellen Eindruck über die durchgeführte Messung (Beispiel siehe Abbildung 1).

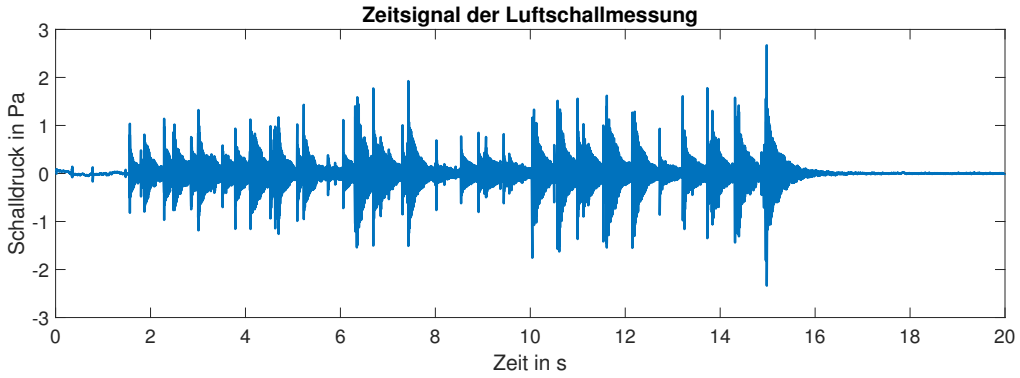

Abbildung 1: Zeitsignal einer durchgeführten Messung (Schallaufzeichnung). Dargestellt wird der Schalldruck in *Pa* über die Zeit in *s*

5 Schalldruckpegelbestimmung mittels SPL-Meter

MATLAB® bietet über das System-Objekt *splMeter* die Funktionalität eines Schallpegelmessgeräts nach [1, DIN EN 61672]. Im Funktionsumfang des *splMeter* sind die Frequenzbewertungen A, C und Z, die Abklingkonstanten fast und slow. Es ist ebenso eine ntel-Oktavanalyse enthalten, welche jedoch besser und umfangreicher über ein anderes MATLAB®-Modul realisiert wird.

```
SPL = splMeter('FrequencyRange', [22 22050], ...
      'FrequencyWeighting', 'A-weighting', ...
      'TimeWeighting', 'Fast', ...
      'PressureReference', 20e-6, ...
      'CalibrationFactor', 1, ...
      'TimeInterval', 1, ...
      'SampleRate', fs);
```

definiert das System-Objekt *splMeter*. Die hier durchgeführte Parametrierung des *splMeter* entspricht der üblichen Einstelllung eines Handschallpegelmessers für A-bewertete Schallpegelmessungen. Als Kalibrierfaktor (CalibrationFactor) ist in diesem Fall 1 zu verwenden, da aufgrund der Besonderheit des Mikrofons die sonst übliche Umrechnung von gemessener Spannung bzw. ermittelter Messbereichsaussteuerung auf den Schalldruckmesswert nicht möglich ist.

Über die Anweisung

```
[Lt,Leq] = SPL(data);
```

erfolgt die Berechnung der Pegelwerte des Momentanpegels $L(t)$ sowie des equivalenten Pegels L_{eq} jeweils als dB(A)-Werte (siehe Abbildung 2).

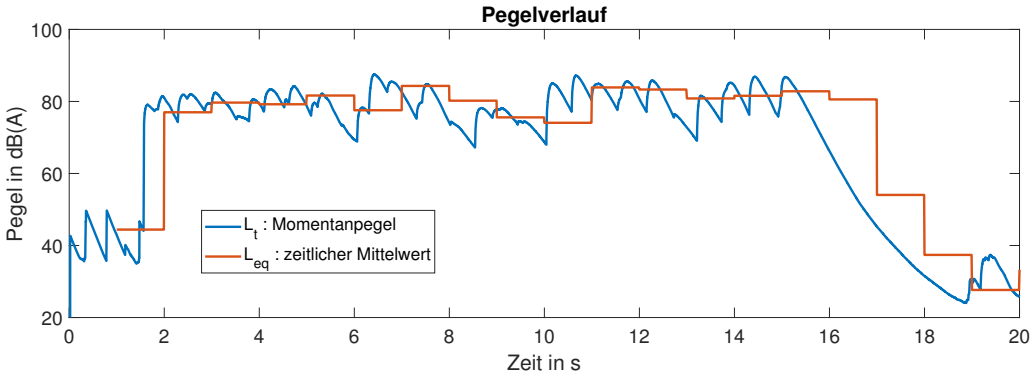

Abbildung 2: Darstellung des Momentanpegels und des equivalenten Pegels einer durchgeführten Schallaufzeichnung. Dargestellt wird der Schalldruckpegel in $dB(A)$ über die Zeit in s

6 Frequenzanalyse mittels Bandpassfilter für Oktaven und Teiloktaven

Unter der in der technischen Umgangssprache als Oktaven und Terzen bezeichneten Frequenzanalyse handelt es sich um einen mehrstufigen Analyseprozess, bei dem ein Zeitsignal in mehrere Zeitsignal-Datenströme aufgeteilt wird. Hierbei findet zunächst eine erhebliche Datenmehrung statt. Erst weiterführende Schritte führen eine Datenreduktion durch. Soll die Frequenzanalyse auf ein A-bewertetes Schallsignal durchgeführt werden, so ist als erster Arbeitsschritt diese Filterung durchzuführen. Die MATLAB® DSP System Toolbox stellt für die A-Bewertung ein Filterdesign zur Verfügung, welches sich als Programmfunktion generieren lässt, so dass lediglich die Anweisung

```
data_A = afilter(data, fs);
```

erforderlich ist, um einen A-bewerteten (A-gefilterten) Zeitdatensatz zu erhalten.

Anhand von n Bandpassfilter werden aus dem Zeitsignal die Analysefrequenzbereiche einzeln herausgefiltert. Bei Oktaven ergeben sich für den hörbaren Frequenzbereich zehn Bandpassfilter. Sind schmälere Frequenzbänder für die Analyse erforderlich als Oktaven werden Teiloktaven gebildet. Die 1/3-Oktave trägt dabei den Namen Terzfilter bzw. Terz. Alle anderen Teilungen werden in der Form 1/n-Oktave bezeichnet.

Gemäß [2, DIN EN 61260] werden die Eck- und Mittenfrequenzen der Teiloktavfilter anhand

$$f_1 = \frac{f_2}{\sqrt[n]{2}} \tag{2}$$

sowie

$$f_0 = \sqrt{f_1 \cdot f_2} = f_1 \cdot \sqrt{\sqrt[n]{2}} \tag{3}$$

bestimmt.

Zu einem Bandpassfilter muß die Mittenfrequenz f_0 definiert sein. Für Luftschallmessungen ist dies $f_0 = 1000\,Hz$.

Die MATLAB® DSP System Toolbox stellt für das Filterdesign der benötigten Teiloktaven-Bandpassfilter ein Filterdesignobjekt zur Verfügung, welches eine Teiloktav-Frequenzanalyse erheblich vereinfacht. Die Anweisung

```
ntel =      6;
filterdefs = fdesign.octave(ntel,'Class 0','N,F0',8, 1000,fs);
```

definiert in diesem Beispiel das Filterdesign für die 60 einzelnen Bandpassfilter, welche für eine 1/6-Oktavanalyse benötigt werden.

Werden die 60 Filter angewendet, so ergeben sich insgesamt 60 Zeitdatenströme, welche eine weitere Signalanalyse benötigen. Üblich wird je Teiloktave ein Gesamtpegel $L(t)$ bestimmt. Was wiederum über das System-Objekt *splMeter* durchgeführt wird. Dies hat eine geringfügig andere Konfiguration als in der Anwendung in Abschnitt 5.

```
SPL = splMeter('FrequencyRange', [22 22050], ...
      'FrequencyWeighting', 'Z-weighting', ...
      'TimeWeighting', 'Fast', ...
      'PressureReference', 20e-6, ...
      'CalibrationFactor', 1, ...
      'SampleRate', fs);
```

Der Pegelverlauf der einzelnen Teiloktaven wird nicht in der zeitlichen Auflösung der Messung benötigt. In diesem Beispiel wären dies 88200 Pegelwerte je Sekunde. Durch geschickte Wahl der Zeitauflösung für den Pegelverlauf lässt sich eine erhebliche Datenreduktion durchführen, ohne dass es zu einem spürbaren Informationsverlust kommt. Benötigt wird dazu eine "neue"Zeitachse, welche durch

```
tAchse = time(1,1):1/1000:time(end,1);
```

erstellt wird. Diese weist eine Zeitauflösung von $1/1000\,s = 1\,ms$ auf. Über die beiden Anweisungen

```
frequenzen =  validfrequencies(filterdefs);
      Nfc =   length(frequenzen);
```

werden die Teiloktav-Mittenfrequenzen und deren Anzahl bestimmt, um darüber die Teiloktav-Filterung bzw. die 1/n-Teil-Oktav-Frequenzanalyse in einer Schleife durchzuführen.

```
for   xi=1:Nfc
      filterdefs.F0 = frequenzen(xi);
      to(:,xi) = interp1(time, SPL(filter(design(filterdefs, 'butter'),data_A)), tAchse);
end
```

In der Matrix to sind danach die n Momentanpegelverläufe $L(t)$ der einzelnen Teiloktave mit einer zeitlichen Auflösung von $\Delta t = 1\ ms$ abgelegt. Über eine der Anweisungen zur Visualisierung von 3D-Datensätzen lässt sich die Frequenzanalyse Visualisieren (siehe Abbildung 3).

```
mesh(frequenzen,tAchse, to);
```

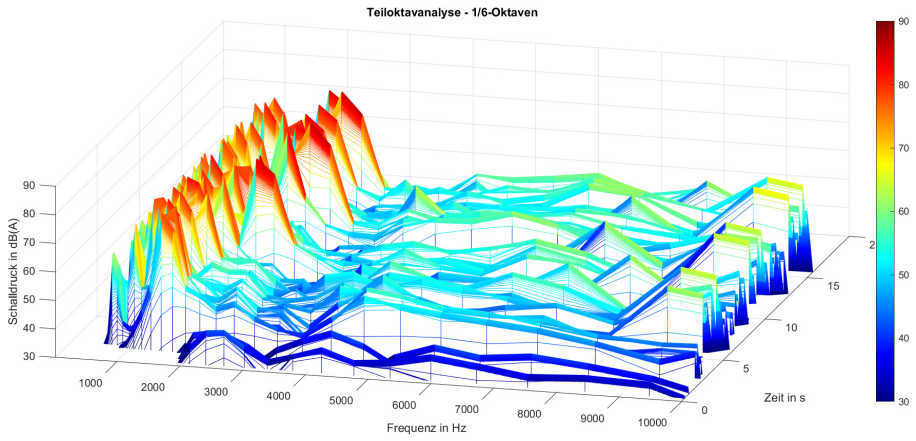

Abbildung 3: Teiloktavanalyse einer durchgeführten Messung (Schallaufzeichnung). Dargestellt wird der Schalldruck der Teiloktaven (16tel-Oktave) in Pa über die Zeit in s

7 Frequenzanalyse mittels Fouriertransformation

Die durchaus üblichste Methode zur Signalanalyse von Schallsignalen ist die Transformation des Signals aus der Zeitebene in die Frequenzebene und Anwendung der Fouriertransformation. Dabei ist es für das Ergebnis der Transformation unerheblich ob sich der Algorithmus der Fast-Fouriertransformation (FFT) oder der diskreten Fouriertransformation (DFT) bedient. Beide Formen der Fouriertransformation stellen die Lösung der Gleichung

$$\underline{X}(n) = \frac{1}{N} \sum_{k=0}^{N-1} x(k)\ e^{-j2\pi \frac{kn}{N}} \tag{4}$$

dar. Der Unterschied besteht in der Abarbeitungsgeschwindigkeit bzw. der Zeitspanne, welche für die Berechnung der Lösung erforderlich ist. Bei üblichen Datenmengen und heute üblichen Rechnerkonfigurationen ist ein zeitlicher Unterschied jedoch nur noch marginal feststellbar. Für die Berechnung von FFTs werden Datenblöcke der Größe 2^N benötigt, was in Verbindung mit den weiteren Parametern der Fouriertransformation und den Abtastraten der Soundkarten allerdings zu ”krummen”Frequenzauflösungen führt.

Die Durchführung einer Fouriertransformation benötigt gemäß [3, Thomas Kuttner, Armin Rohnen] zwei individuelle Entscheidungen. Es ist die maximal erforderliche Analysefrequenz zu definieren sowie die

Linienbreite bzw. Frequenzauflösung festzulegen. Anhand der maximalen Analysefrequenz f_{max} ergibt sich die erforderliche Abtastrate für die Datenerfassung

$$f_s \geq f_{max} \cdot 2,56 \tag{5}$$

In der Praxis wird die Messung mit der nächsthöheren Abtastrate f_s durchgeführt. Aufgrund des Grundgesetztes der Nachrichtentechnik, der Unschärferelation,

$$T \cdot \Delta f = 1 \tag{6}$$

bestimmt die Definition der Frequenzauflösung Δf die Blocklänge T und damit die Mittelungszeit der Fouriertransformation.

Die Linienanzahl N, auch Blockgröße bzw. Anzahl der Abtastwerte, ergibt sich aus dem Verhältnis der Abtastrate und der Frequenzauflösung

$$N = \frac{f_s}{\Delta f}. \tag{7}$$

Es ist darauf zu achten, dass N zwingend ein ganzzahliger Wert sein muß.

Zur Reduktion des Leakage wird der Datenblock für die Fouriertransformation mit einer Fensterfunktion $w(n)$ multipliziert. Üblicher weise wird die Hanning-Fensterfunktion dazu verwendet. Aus Gründen der Vergleichbarkeit sollte dies beibehalten werden.

Die Verwendung einer Fensterfunktion führt zu einem Fehler in der Betragsbetrachtung, welches durch die Korrektur mittels

$$FM = \frac{\sum w(n)}{N} \tag{8}$$

behoben wird.

Da bei der Verwendung einer Fensterfunktion zu Beginn und am Ende der Datenblöcke jeweils Datenlücken entstehen, sollten die Datenblöcke für die Fouriertransformation nicht Block-an-Block sondern mit einer Überlappung verwendet werden. Ideal ist dabei eine Überlappung (overlap) von

$$o = \frac{2}{3} \cdot N \tag{9}$$

Mit *spectrogram* stellt MATLAB® eine komfortable Anweisung für die Durchführung einer Fouriertransformation zur Verfügung. Dabei ist es unerheblich, ob die Daten für die Transformation in 2^N vorliegen oder nicht. Es wird je nach Blockgröße dynamisch entschieden, ob eine FFT oder DFT erstellt wird. Für das numerische Ergebnis ist dies eh zweitrangig.

```
df =        4;
N =         ceil(fs/df);
o =         ceil(2/3*N);
w =         hann(N);
FM =        sum(w)/N;
data_A =    afilter(data, fs);

[s,f,t] =   spectrogram(data_A, w, o, N, fs);
```

führt eine Fouriertransformation der Daten *data* mit einer Frequenzauflösung $\Delta f = 4\ Hz$ durch. Als Ergebnis der Transformation steht in s die Matrix der nicht normierten, komplexen Fourierkoeffizienten, in f der Vektor der Frequenzachse und in t der Vektor der Zeitachse zur weiteren Verarbeitung und zur Visualisierung zur Verfügung.

Die Fourierkoeffizienten liegen bereits auf Einseitigkeit reduziert vor. Jedoch wurde die Normierung (Mittelwertbildung) noch nicht durchgeführt. Ebenso fehlt die Amplitudenkorrektur an die verwendete Fensterfunktion. Über

```
Sx = 2*abs(s/N)/FM;
```

wird die Matrix der Betragsspektren ermittelt, welche über

```
mesh(f',t',20*log10(Sx'/(20e-6)));
```

in Abbildung 4 dargestellt ist.

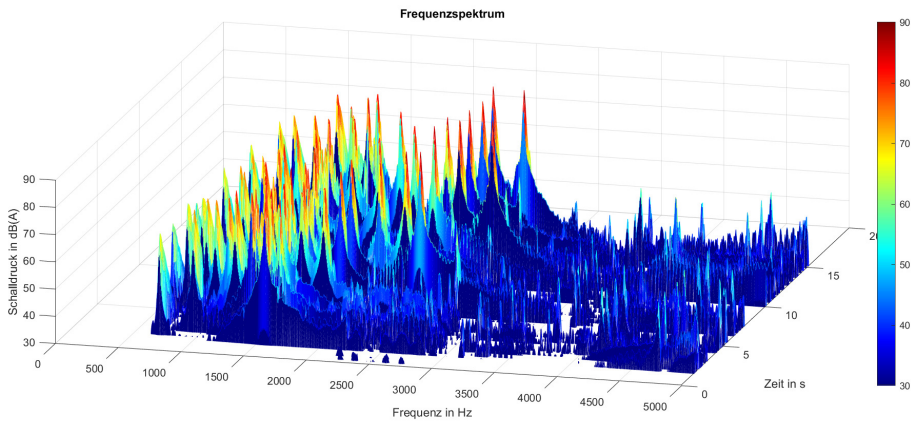

Abbildung 4: Frequenzspektrogramm einer durchgeführten Messung (Schallaufzeichnung). Dargestellt wird der frequenzabhängige Schalldruck in $dB(A)$ über die Zeit in s

Auch Frequenzschnitte (Beispiel siehe Abbildung 5) und Zeitschnitte (Beispiel siehe Abbildung 6) lassen sich mittels der Matrix der Betragsspektren darstellen.

Beim Freqenzschnitt wird dazu der Index der Frequenzlinie für die Darstellung benötigt.

```
fView =   1400;
nView =   ceil(fView/df)+1;
```

bestimmt z. B. den Index der Frequenzlinie für den Frequenzschnitt bei $f = 1400\ Hz$. Der berechnete Index ist um 1 zu erhöhen, da MATLAB® über keinen Index 0 verfügt und dadurch den Index um $+1$ versetzt. Die Anweisung

```
plot(t, 20*log10(Sx(nView, :)/20e-6))
```

führt danach die Visualisierung durch.

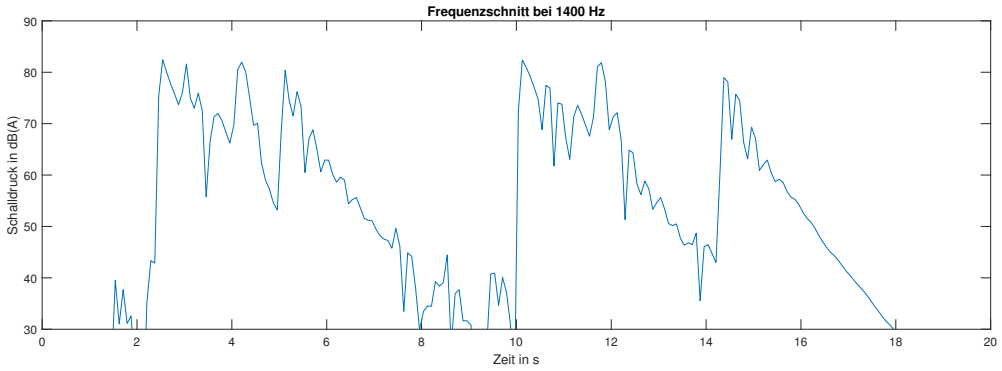

Abbildung 5: Schnitt entlang der Frequenzachse durch das Frequenzspektrogramm bei $f = 1400\ Hz$. Dargestellt wird der Schalldruck in $dB(A)$ über die Zeit in s

Analog zum Frequenzschnitt wird der Zeitschnitt durchgeführt. Hier wird der Index für die Zeitlinie in der Matrix über

```
tView =   10.3;
nView =   ceil(tView/(t(1,10)-t(1,9)))+1;
```

bestimmt und mittels

```
plot(f, 20*log10(Sx(:, nView)/20e-6))
```

die Darstellung durchgeführt.

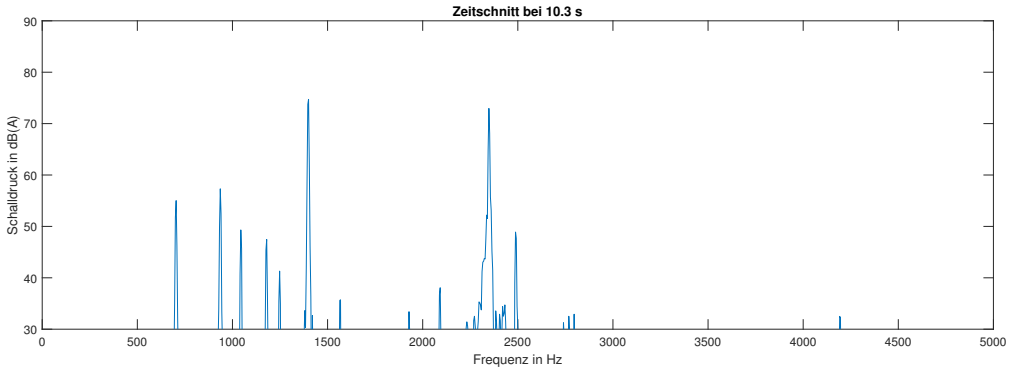

Abbildung 6: Schnitt entlang der Zeitachse durch das Frequenzspektrogramm bei $t = 10,3\ s$. Dargestellt wird der Schalldruck in $dB(A)$ über die Frequenzen s

8 Zusammenfassung

Das digitale Mikrofon MV 240 wird auf einem Windows PC als Soundkarte identifiziert und ein Datenzugriff auf den einkanaligen USB-Datenstrom in 32 Bit Wortbreite ist mit wenigen Zeilen MATLAB®-Programmcode möglich. Der im Mikrofonverstärker enthaltene Signalprozessor normiert den Datenstrom auf eine Signalauflösung von 1 μPa.

In diesem Dokument sind neben den MATLAB®-Anweisungen für die Signalaufzeichnung die Toolboxen für die üblichen Signalanalysen von Luftschallsignalen dargelegt. Neben der Schallpegelberechnung werden Signalanalysen mittels Teiloktavfilterung und Fouriertransformation dargelegt.

9 Literatur und Quellen

Literatur

[1] DIN EN 61672, Elektroakustik, Schallpegelmesser

[2] DIN EN 61260, Elektroakustik, Bandfilter für Oktaven und Bruchteile von Oktaven

[3] Thomas Kuttner, Armin Rohnen, Praxis der Schwingungsmessung, Messtechnik und Schwingungsanalyse mit MATLAB®, 2. Auflage, Springer Vieweg, Wiesbaden, 2019