

Sprachverarbeitung: Übung 2

Quantisierung von Sprachsignalen

Aufgabe 1: Aus Sinuswellen zusammengesetzte Signale

Schreiben Sie ein Matlab-Skript, welches unter Verwendung der Funktion `logswEEP`¹ die folgenden, je drei Sekunden langen Signale mit einer Abtastfrequenz von 8000 Hz erzeugt:

- Ein harmonisches Signal mit den ersten sieben Oberwellen, wobei die Grundwelle eine Anfangsfrequenz von 200 Hz, eine Endfrequenz von 500 Hz und eine Amplitude von 0.1 haben soll. Das Amplitudenverhältnis der $n+1$ -ten zur n -ten Oberwelle soll 0.7 betragen.
- Dasselbe Signal wie unter a), jedoch ohne Grundwelle.
- Ein Signal mit N nicht-harmonischen Sinuskomponenten. Mit der Funktion `logswEEP` können Sie ein solches Signal generieren, indem Sie für die Anfangs- und Endfrequenzen und für die Anfangsphase Zufallswerte einsetzen (Funktion `rand`). Setzen Sie die Amplitude aller Komponenten auf $0.1/\sqrt{N}$. Wählen Sie N zwischen 10 und 500.

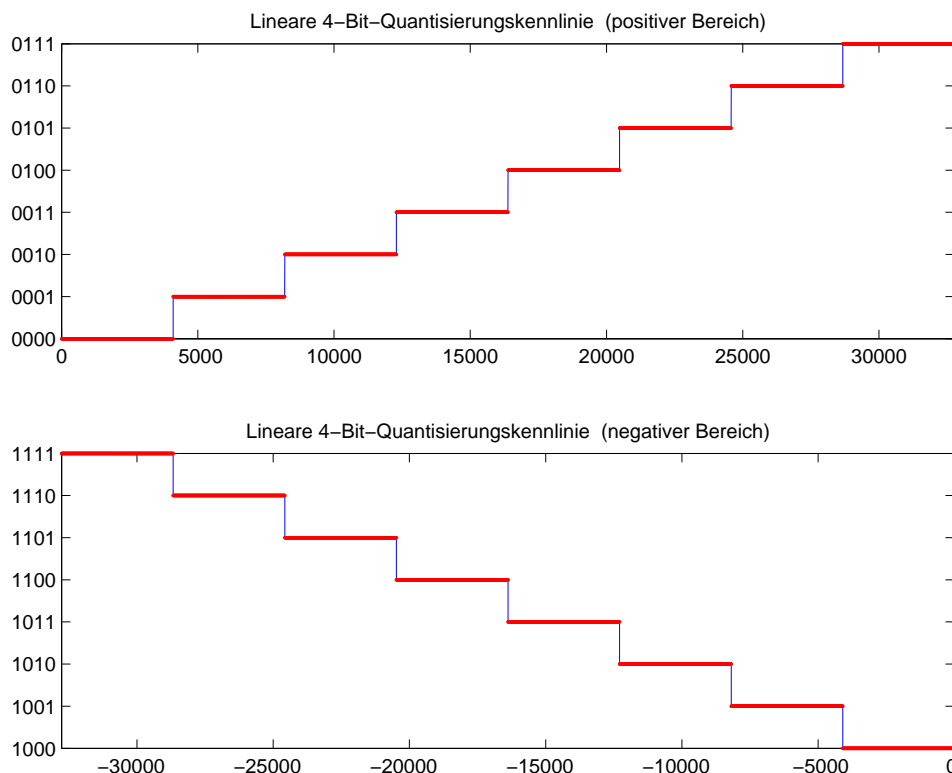
Was stellen Sie beim Visualisieren bzw. Anhören dieser Signale fest? Benutzen Sie zur graphischen Darstellung der ersten n Abtastwerte des Signals x die Funktionen `plot(x(1:n))`. Zum Anhören des Signals x können Sie die Funktion `sound(x,8000)` benutzen.

Aufgabe 2: Quantisierung eines Sprachsignals

Unter einer N -Bit-Quantisierung der Abtastwerte eines Signals versteht man die zwei folgenden Schritte:

- Codierung: Der Wertebereich ist in 2^N Intervalle unterteilt und jedem Intervall ist ein N -Bit-Code zugeordnet, wie dies in Figur 1 für einen Wertebereich von $-2^{15} \dots + 2^{15}$ und einen 4-Bit-Code dargestellt ist. Einen Signalabtastwert s zu codieren heisst also, zu ermitteln in welchem Intervall s liegt und den zugehörigen N -Bit-Code c ausgeben.
- Decodierung: Es wird für den N -Bit-Code c der entsprechende Wert \tilde{s} ermittelt. Damit die Abweichung zwischen s und \tilde{s} möglichst gering ist, müssen die Werte von \tilde{s} in der Mitte der Quantisierungsintervalle liegen.

¹Die Matlab-Funktion `logswEEP` verlangt die Eingabe der Anfangs- und der Endfrequenz in normierter Form, also als f/f_s , wobei f_s die Abtastfrequenz ist.



Figur 1: Kennlinie der linearen 4-Bit-Quantisierung für den positiven Eingangswertebereich ($0 \dots 2^{15}$) oben und für den negativen ($-2^{15} \dots -1$) unten. Jedem Quantisierungsintervall ist ein 4-Bit-Code zugeordnet. Merke: Im resultierenden 4-Bit-Code ist das erste Bit das Vorzeichen, die restlichen Bits geben den Betrag an (kein 2^{er} -Komplement!).

Realisieren Sie ein Matlab-Programm, mit dem ein gegebenes Sprachsignal wahlweise linear oder logarithmisch quantisiert (Codierung und Decodierung) werden kann. Das in der Datei `ueb2_2_sig.wav` vorliegende Sprachsignal ist mit 8 kHz und 16 Bit digitalisiert worden. Beachten Sie für die Realisation des Programms folgende Hinweise:

- Um den Programmieraufwand in Grenzen zu halten, können Sie ihre Lösung auf dem Matlab-Rahmenprogramm `ueb2_2_frame.m` aufbauen.
- Zum Laden des Signals können die beiden folgenden Befehle (sind im Rahmenprogramm bereits aufgeführt) verwendet werden:

```
[sig16, SamplFreq] = audioread('ueb2_2_sig.wav', 'native');
sig = double(sig16);
```

Der erste Befehl liest aus der Wave-Datei die 16-Bit-Abtastwerte und gibt sie als ebensolche im Vektor `sig16` aus. Da mit dem Matlab-Datentyp "int16" nicht allgemein gerechnet werden kann, wird das Signal mit dem zweiten Befehl in den Datentyp "double" umgewandelt. Das Signal ist sodann im Vektor `sig` verfügbar.

Abfragen, mit welcher Auflösung die Abtastwerte in der Wave-Datei gespeichert sind, kann man wie folgt:

```
ainfo = audioinfo('ueb2_2_sig.wav');
NB = ainfo.BitsPerSample;
```

Dabei bezeichnet NB die Anzahl Bits pro Abtastwert.

- Für einen Signalabtastwert **sig** mit einer Auflösung von **NB** Bits kann wie folgt der einer linearen Quantisierung entsprechende Code mit **NBcode** Bits ermittelt werden (sinnvollerweise ist $NB > NBcode$):

```
scal = 2^(NBcode-NB);
codlin = floor(scal*min(abs(sig),2^(NB-1)-1)) + 2^(NBcode-1)*(sig<0);
```

Der Skalierungsfaktor ist so angesetzt, dass der Betrag der ganzzahligen EingangsgröÙe im Bereich $0 \dots 2^{15}-1$ auf den Wertebereich $0 \dots 2^{NBcode-1}-1$ transformiert wird. Die resultierenden Werte werden ganzzahlig gemacht und für negative Werte wird das Vorzeichen (MSB des **NBcode** Bits langen Codewortes) auf 1 gesetzt. Wenn Sie diese Codierung verstanden haben, dann tragen Sie sie ins Matlab-Skript ein und testen Sie, ob die Quantisierungskennlinie (Matlab Figur 3) der Erwartung entspricht (**TestMode** = 1 setzen).

- Überlegen Sie, wie die Decodierung (Ermitteln des quantisierten Signalabtastwertes im Bereich $-2^{15} \dots +2^{15}-1$ aus dem Code) gemacht werden muss, damit die decodierten Werte in der Mitte der Quantisierungsintervalle zu liegen kommen und somit der Quantisierungsfehler minimal bzw. das SNR^2 maximal wird. Im Test-Mode werden die decodierten Werte in der Quantisierungskennlinie (Matlab Figur 3) als \otimes eingetragen.
- Wenn Ihre Decodierung korrekt funktioniert, können Sie **TestMode** = 0 setzen und die lineare Quantisierung auf das Sprachsignal anwenden und das Ergebnis anhören.
- Für die logarithmische Quantisierung wird der Betrag der Abtastwerte einer Transformation der folgenden Form unterworfen:

$$y = a \cdot \log(x) \quad (1)$$

Der Faktor a wird so eingestellt, dass der Wertebereich von x auf den Wertebereich von y abgebildet wird, also $a = y_{max}/\log(x_{max})$. Dabei ist jedoch zu berücksichtigen, dass die Transformation vom Wertebereich von x abhängt. Figur 2 zeigt, wie unterschiedlich die Transformation für die Wertebereiche $x = [1 \dots 100]$ und für $x = [1 \dots 10000]$ ausfallen. Der Effekt des Wertebereiches kann eliminiert werden, indem zur Transformation die Funktion

$$y = a \cdot \log(x \cdot d) = y_{max}/\log(x_{max} \cdot d) \cdot \log(x \cdot d) \quad (2)$$

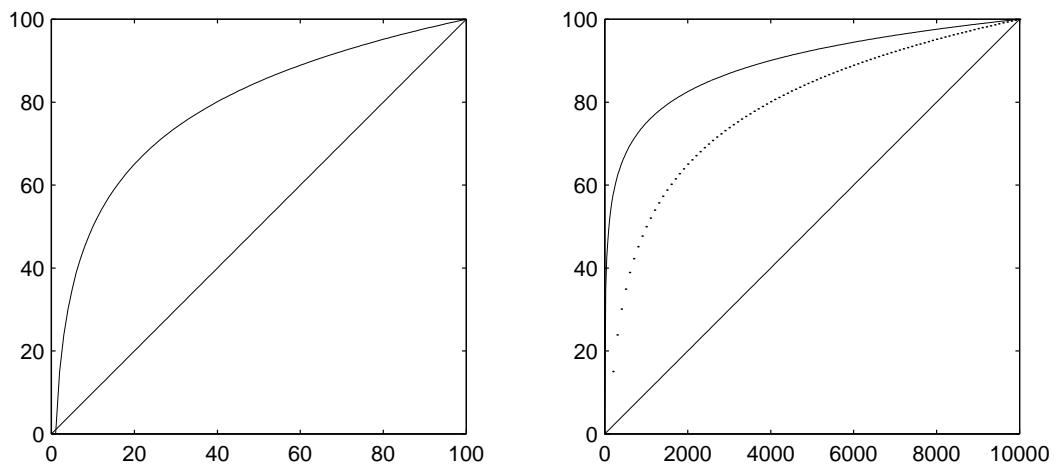
verwendet wird, wobei der Parameter d empirisch so zu bestimmen ist, dass das SNR maximal wird.

Programmieren Sie die Codierung und die Decodierung für die logarithmische Quantisierung. Um zu verhindern, dass $\log(x)$ für kleine Werte negativ wird, verwenden Sie $\log(\max(x, 1))$. Überlegen Sie, warum dies sinnvoll ist.

Lassen Sie das Programm zuerst im Test-Mode laufen und überprüfen Sie anhand der Quantisierungskennlinien in der Matlab-Figur 5, ob die Quantisierung korrekt ist.

Vergleichen Sie das SNR für verschiedene Codelängen $K = 1 \dots 8$ bei linearer bzw. logarithmischer Quantisierung. Vergleichen Sie die decodierten Signale auch akustisch. Beachten Sie, dass bei der Funktion **sound** der Wertebereich des Signals $-1 \dots +1$ sein muss.

²Das Verhältnis von Signal- zu Rauschleistung (engl. *signal-to-noise ratio* oder kurz SNR) wird in dB angegeben und wird mit $SNR = 10 \log_{10}(\sum_n s(n)^2 / \sum_n r(n)^2)$ berechnet, wobei $s(n)$ das Signal und $r(n)$ das Rauschen ist.



Figur 2: Lineare und logarithmische Transformation für verschiedene Eingangswertebereiche bei gleichem Ausgangswertebereich. Die punktierte Kurve rechts, die mit der Form der Kennlinie in der linken Figur übereinstimmt, ergibt sich aus der Funktion (2) mit $d = 0.01$.