HOCHSCHULE EMDEN·LEER

Institut für Medien und Technik

Codierung multimedialer Daten

Aufgaben zum nächsten Mal (AZNM 9)

Johann-Markus Batke

2023-05-31

1 Quellencodierung

Machen Sie sich mit folgenden Begriffen und Definitionen vertraut:

- https://vfhcmd.eduloop.de/loop/Differenzcodierung
- https://vfhcmd.eduloop.de/loop/Differenzen_Entropie
- https://vfhcmd.eduloop.de/loop/Differenzen:_Quantisierung

Zu einigen Aufgaben finden Sie auch gleich die Lösung als Implementierung in Python. Nehmen Sie diese als Anregung für eine eigene Umsetzung. Variieren Sie die gegebenen Parameter.

2 Aufgaben

2.1 Differenzcodierer

Zeichnen Sie Encoder und Decoder eines Differenzencodierers mit $x[n]$ als Eingangssignal und $d[n]$ als Differenzsignal.

2.2 Differenzcodierung

Bilden Sie den Differenzcode der Zahlenfolge

30 -50 51 49 61 55 101 138 43 8 28 28 28 28 28 28

Wählen Sie als Initialisierungswert des Codecs

0

2.3 Differenzcodierung/Entropie

Wie können Sie das Ergebnis der Differenzcodierung in Hinsicht auf eine datenreduzierende Codierung interpretieren?

2.4 Experiment

Bestimmen Sie die Entropie eines selbstausgewählten Musiksignals. Führen Sie die Differenzcodierung durch und bestimmen Sie die Entropie erneut!

2.4.1 Loesung

init.

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.io.wavfile as wavfile
```

Musik-Datei einlesen:

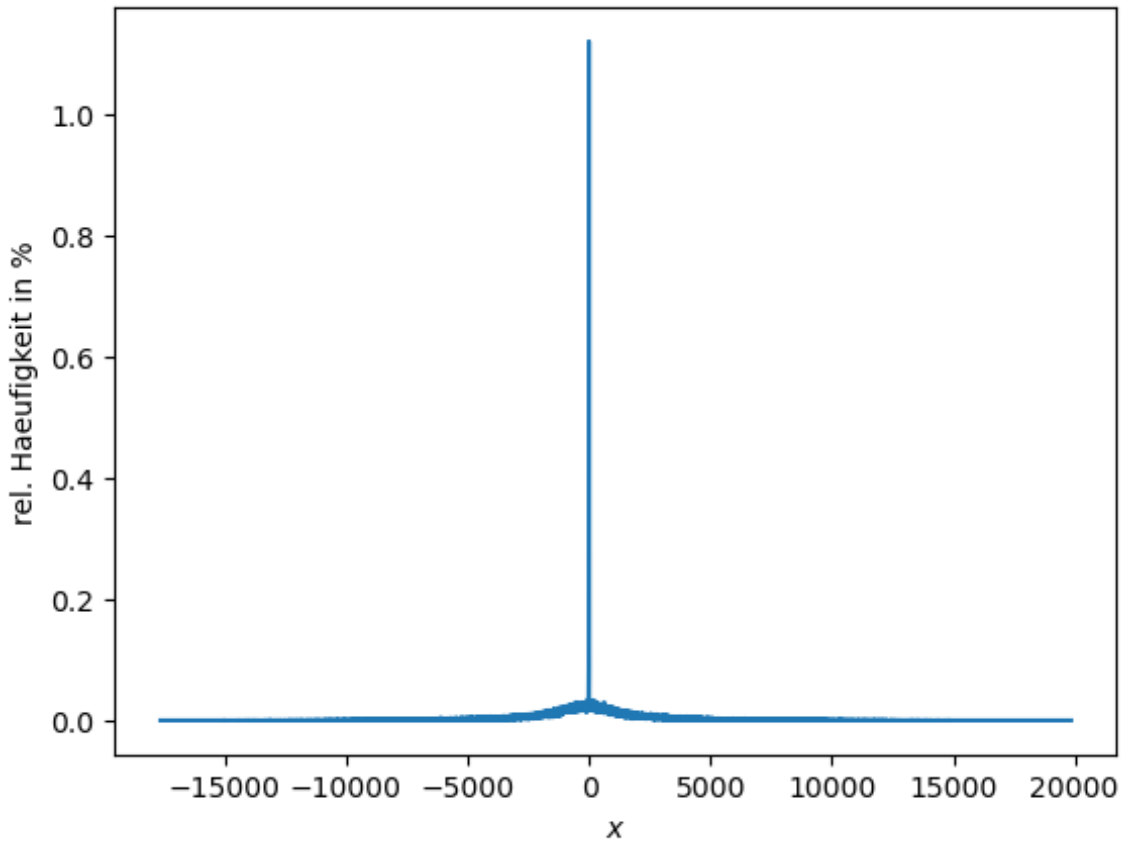
```
S_filename = "../Vorlesung/1Battle_Royal.wav"
[f_rate, x_data] = wavfile.read(S_filename)
x_L = np.array(x_data, dtype=float)
plt.plot(x_L)
```

WDF bestimmen:

Schätzung der Wahrscheinlichkeitsdichtefunktion $f_{\boldsymbol{x}}(x)$ per Häufigkeitsdichte

```
# x_data = np.random.lognormal(0, 1, 10000)
bins_x = np.arange(np.min(np.min(x_data)), np.max(np.max(x_data))+1)
f_1_x, bin_edge = np.histogram(x_data[:,0], bins_x, density=True)
plt.plot(bins_x[1:bins_x.size], f_1_x*100)
plt.xlabel("$x$")
plt.ylabel("rel. Haeufigkeit in %")
```

```
Text(0, 0.5, 'rel. Haeufigkeit in %')
```



Entropie $H(x) = \sum_i P_{\mathbf{x}}(x_i) \log_2(\frac{1}{P_{\mathbf{x}}(x_i)})$

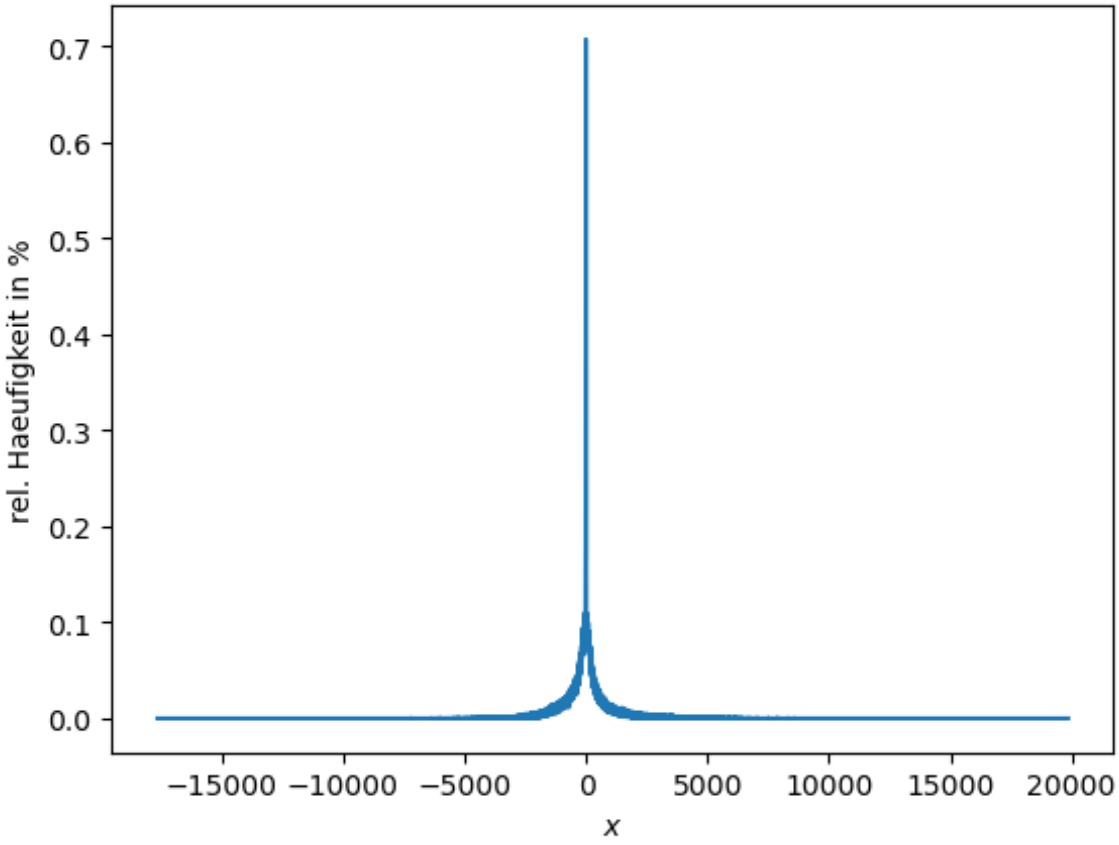
```
def entropie(f_x):
   nz = np.where(f_x != 0)
   return -np.sum(f_x[nz] * np.log2(f_x[nz]))
print("Entropie H_x = ", entropie(f_1_x), "bit")
```

Entropie H_x = 12.89180477512903 bit

WDF für $d[n]$

```
d_L = np.diff(x_L[:,0])
f_1_d, bin_edge = np.histogram(d_L, bins_x, density=True)
plt.plot(bins_x[1:bins_x.size], f_1_d*100)
plt.xlabel("$x$")
plt.ylabel("rel. Haeufigkeit in %")
```

```
Text(0, 0.5, 'rel. Haeufigkeit in %')
```



Entropie des Differenzsignals

```
print("Entropie H_d = ", entropie(f_1_d), "bit")
```

Entropie H_d = 11.436124159935002 bit

Die Ersparnis durch die Differenzcodierung beträgt etwa

```
print(entropie(f_1_x)-entropie(f_1_d), " bit")
```

1.4556806151940282 bit

2.5 Differenzen-Quantisierung

Nachfolgend finden Sie die Angaben aus der Exceltabelle des Abschnitts. Vollziehen Sie die Quantisierung und Synthese des Ursprungssignals nach!

Die Grenzwerte der ungleichförmigen (und linearen) Quantisierung (nach der Tabelle https://vfhcmd.eduloop.de/loop/Differenzen:_Quantisierung) sind in Abbildung 1 dargestellt, die

```
Q_Gr = np.array([-10000, -26.5, -11.5, -4.5, -1.5, -0.5])
Q_Gr = np.concatenate((Q_Gr,  -Q_Gr[Q_Gr.size::-1]))
Q_Er = np.array([-38, -19, -8,  -3, -1, 0])
Q_Er = np.concatenate((Q_Er, -np.flip(Q_Er[:-1], axis=0)))
```

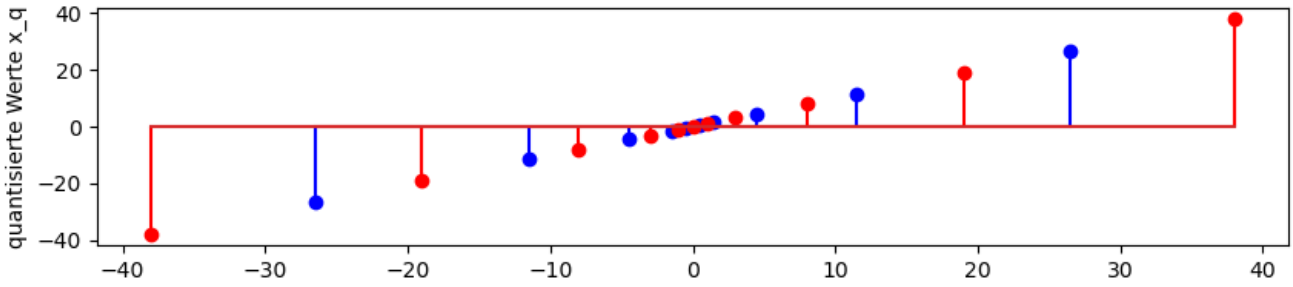


Abbildung 1: Werte zur ungleichförmigen Quantisierung.

Definiere Quantisierer

```
def quantisierer(x_lin_N, Q_Gr, Q_Er):
    x_qun_N = np.zeros(x_lin_N.size)

    for g in range(Q_Er.size): # suche für jeden Ersatzwert Kandidaten
        idx = np.logical_and(x_lin_N >  Q_Gr[g], x_lin_N <= Q_Gr[g+1]) # untere Grenze # obere Grenze
        x_qun_N[idx] = Q_Er[g] # weise Ersatzwert zu
    return x_qun_N
```

Die Darstellung der Kennlinie über linear aufsteigende Eingangswerte ist in Abbildung 2 dargestellt.

Fehler zwischen den Signalen

```
e_qun_N = d_qun_N - d_N
```

und kumulierter Fehler

```
e_ku_N = np.cumsum(e_qun_N)
```

synth. Signal aus d_{qnN} und d_N

```
X_qun_N = dpcm_dec(d_qun_N)
X_N = dpcm_dec(d_N)
```

Abbildung 3 zeigt die Darstellung der beiden Varianten und den kumulierten Fehler:

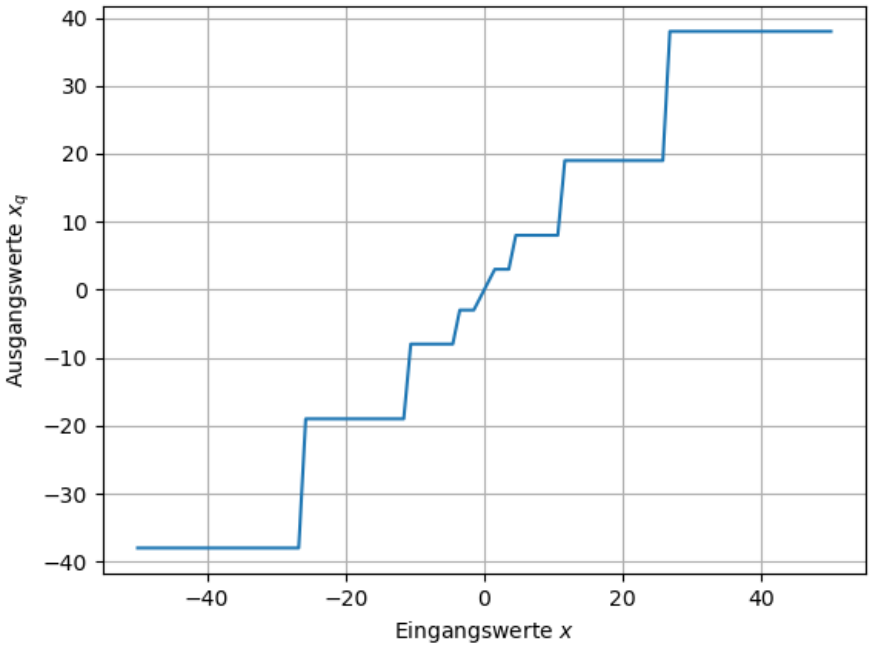


Abbildung 2: Kennlinie des Quantisierers.

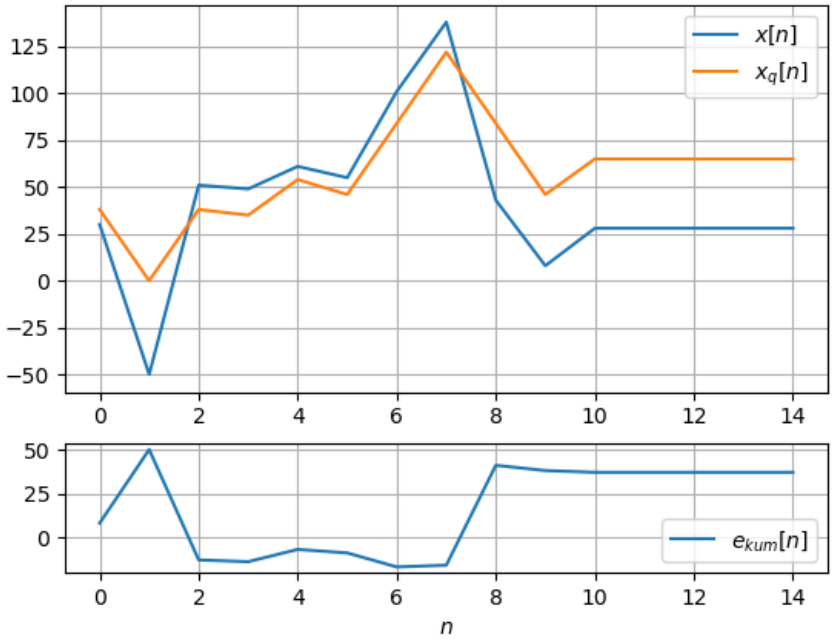


Abbildung 3: Darstellung des synthetisierten Ausgangssignals aus quantisierten und nicht quanitisierten Differenzen. Darunter der kumulierte Fehler.