REPORT

Zajęcia: Analog and digital electronic circuits

Teacher: prof. dr hab. Vasyl Martsenyuk

Lab 7

Date: 5.04.2024

Topic: "Quantization and Dithering"

Variant: 13

Agnieszka Białecka

Informatyka II stopień,

stacjonarne,

1 semestr,

Gr.1a

1. Abstract

The objective was to investigate dithering technique for different signals.

2. Theoretical introduction

The discrete-time sine signal

- $x[k] = A \cdot \sin(\frac{2\pi f_{sin}}{f_s}k)$ for - 0 ≤ k < 96000 with - sampling frequency fs $= 48$ kHz and

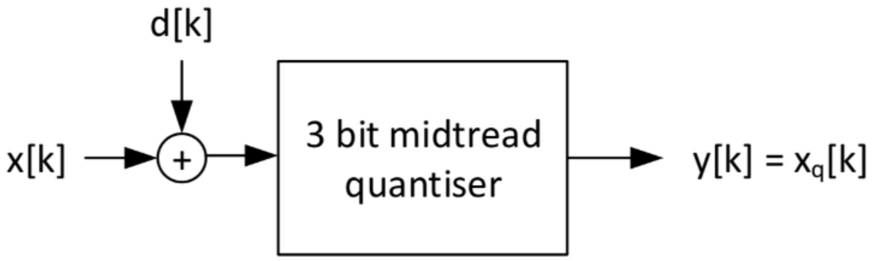
- fsin $= 997$ Hz

shall be quantized with the saturated uniform midtread quantizer for −1 ≤ xq ≤ 1 − ΔQ using B bits, i.e. Q = 2B number of quantization steps

and quantization step size of $\Delta Q = \frac{1}{Q\backslash 2}$.

We should discuss different parametrizations for signal amplitude A and number of bits B.

Before quantizing x[k], a dither noise signal d[k] shall be added to x[k] according figure below.



This dither signal with small amplitudes aims at de-correlating the quantization error e[k] from the quantized signal xq[k], which is especially important for small amplitudes of x[k]. This technique is called dithering. Ford[k] = 0 no dithering is applied.

Since the quantization error may be in the range $-\frac{\Delta Q}{2} \leq e[k] \leq \frac{\Delta Q}{2}$ assuming uniform distribution), it appears reasonable to use a dither noise with a probability density function (PDF) of

$$p_{\mathrm{RECT}}(d) = \frac{1}{\Delta Q} \operatorname{rect}\left(\frac{d}{\Delta Q}\right),$$

i.e. a zero-mean, uniformly distributed noise with maximum amplitude $|d[k]| = \frac{\Delta Q}{2}$. It can be shown that this dither noise improves the quality of the quantized signal. However, there is still a noise modulation (i.e. a too high correlation between xq[k] and e[k]) that depends on the amplitude of the input signal.

The noise modulation can be almost completely eliminated with a zero-mean noise signal exhibiting a symmetric triangular PDF:

$$p_{\mathrm{TRI}}(d) = \frac{1}{\Delta Q}\,\mathrm{tri}\left(\frac{d}{\Delta Q}\right)$$

with maximum amplitude |d[k]| = Q. By doing so, an almost ideal decorrelation between xq[k] and e[k] is realized. In audio, this technique is called TPDF-Dithering (Triangular Probability Density Function Dithering) and can be applied in the mastering process of audio material that is to be dis- tributed e.g. on a CD or via streaming.

3. Input data (Variant)

This report was created with base of variant 13:

fsin	fs
500	1800

GitHub repository:

https://github.com/Delisolara/AaDEC

4. Course of actions

```
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
from scipy import signal
import soundfile as sf
```

Picture 1. Uploaded libraries

```python
def my_quant(x, Q):
    r"""Saturated uniform midtread quantizer

    input:
    x  input signal
    Q  number of quantization steps
    output:
    xq quantized signal

    Note: for even Q in order to retain midtread characteristics,
    we must omit one quantization step, either that for lowest or the highest
    amplitudes. Typically the highest signal amplitudes are saturated to
    the 'last' quantization step. Then, in the special case of log2(N)
    being an integer the quantization can be represented with bits.
    """
    tmp = Q//2  # integer div
    quant_steps = (np.arange(Q) - tmp) / tmp  # we don't use this

    # forward quantization, round() and inverse quantization
    xq = np.round(x*tmp) / tmp
    # always saturate to -1
    xq[xq < -1.] = -1.
    # saturate to ((Q-1) - (Q\2)) / (Q\2), note that \ is integer div
    tmp2 = ((Q-1) - tmp) / tmp  # for odd N this always yields 1
    xq[xq > tmp2] = tmp2
    return xq
```

Picture 2. Implemented code

```python
# cross corelation function
def my_xcorr2(x, y, scaleopt='none'):
    r""" Cross Correlation function phixy[kappa] -> x[k+kappa] y

    input:
    x   input signal shifted by +kappa
    y   input signal
    scaleopt   scaling of CCF estimator
    output:
    kappa   sample index
    ccf   correlation result
    """
    N = len(x)
    M = len(y)
    kappa = np.arange(0, N+M-1) - (M-1)
    ccf = signal.correlate(x, y, mode='full', method='auto')
    if N == M:
        if scaleopt == 'none' or scaleopt == 'raw':
            ccf /= 1
        elif scaleopt == 'biased' or scaleopt == 'bias':
            ccf /= N
        elif scaleopt == 'unbiased' or scaleopt == 'unbias':
            ccf /= (N - np.abs(kappa))
        elif scaleopt == 'coeff' or scaleopt == 'normalized':
            ccf /= np.sqrt(np.sum(x**2) * np.sum(y**2))
        else:
            print('scaleopt unknown: we leave output unnormalized')
    return kappa, ccf
```

Picture 3. Implemented code

```python
# saturation
def check_my_quant(Q):
    N = 5e2
    x = 2*np.arange(N)/N - 1
    xq = my_quant(x, Q)
    e = xq - x

    plt.plot(x, x, color='C2', lw=3, label=r'$x[k]$')
    plt.plot(x, xq, color='C3', label=r'$x_q[k]$')
    plt.plot(x, e, color='C0', label=r'$e[k] = x_q[k] - x[k]$')
    plt.xticks(np.arange(-1, 1.25, 0.25))
    plt.yticks(np.arange(-1, 1.25, 0.25))
    plt.xlabel('input amplitude')
    plt.ylabel('output amplitude')
    if np.mod(Q, 2) == 0:
        s = ' saturated '
    else:
        s = ' '
    plt.title(
        'uniform'+s+'midtread quantization with Q=%d steps, $\Delta Q$=%4.3e' % (Q, 1/(Q//2)))
    plt.axis('equal')
    plt.legend(loc='upper left')
    plt.grid(True)
```

Picture 4. Implemented code

```python
# dithering
def check_dithering(x,dither,Q,case):
    deltaQ=1/(Q//2)#generalrule
    #dithernoise
    pdf_dither, edges_dither=np.histogram(dither,bins='auto', density=True)
    xd=x+dither
    #quantization
    xq=my_quant(xd,Q)
    e=xq-x
    pdf_error,edges_error=np.histogram(e,bins='auto', density=True)
    #writewavs
    sf.write(file='x_'+case+'.wav',data=x,
    samplerate=48000,subtype='PCM_24')
    sf.write(file='xd_'+case+'.wav',data=xd,
    samplerate=48000,subtype='PCM_24')
    sf.write(file='xq_'+case+'.wav',data=xq,
    samplerate=48000,subtype='PCM_24')
    sf.write(file='e_'+case+'.wav',data=e,
    samplerate=48000,subtype='PCM_24')
    #CCF
    kappa,ccf=my_xcorr2(xq,e,scaleopt='biased')
    plt.figure(figsize=(12,3))
    if case=='nodither':
        plt.subplot(1,2,1)
        #nothingtoplotforthezerosignal
        #thePDFwouldbeaweightedDiracatamplitudezero
```

Picture 5. Implemented code

```
  else:
      #plotdithernoisePDFestimateashistogram
      plt.subplot(1,2,1)
      plt.plot(edges_dither[:-1],pdf_dither,'o-',ms=5)
      4
      plt.ylim(-0.1,np.max(pdf_dither)*1.1)
      plt.grid(True)
      plt.xlabel(r'$\theta$')
      plt.ylabel(r'$\hat{p}(\theta)$')
      plt.title('PDF estimate of dither noise')
```

Picture 6. Implemented code

```
#ploterrornoisePDFestimateashistogram
plt.subplot(1,2,2)
plt.plot(edges_error[:-1],pdf_error,'o-',ms=5)
plt.ylim(-0.1,np.max(pdf_error)*1.1)
plt.grid(True)
plt.xlabel(r'$\theta$')
plt.ylabel(r'$\hat{p}(\theta)$')
plt.title('PDF estimate of error noise')
#plotsignals
plt.figure(figsize=(12,3))
plt.subplot(1,2,1)
plt.plot(k,x,color='C2',label=r'$x[k]$')
plt.plot(k,xd,color='C1',label=r'$x_d[k]=x[k]+dither[k]$')
plt.plot(k,xq,color='C3',label=r'$x_q[k]$')
plt.plot(k,e,color='C0',label=r'$e[k]=x_q[k]-x[k]$')
plt.plot(k,k*0+deltaQ,':k',label=r'$DeltaQ$')
plt.xlabel('k')
plt.title('signals')
plt.xticks(np.arange(0,175,25))
plt.xlim(0,150)
plt.legend(loc='lower left')
plt.grid(True)
#plotCCF
plt.subplot(1,2,2)
plt.plot(kappa,ccf)
plt.xlabel(r'$\kappa$')
plt.ylabel(r'$\varphi_{xq,e}[\kappa]$')
plt.title('CCF between xq and e=xq-x')
plt.xticks(np.arange(-100,125,25))
plt.xlim(-100,100)
plt.grid(True)
```

Picture 7. Implemented code

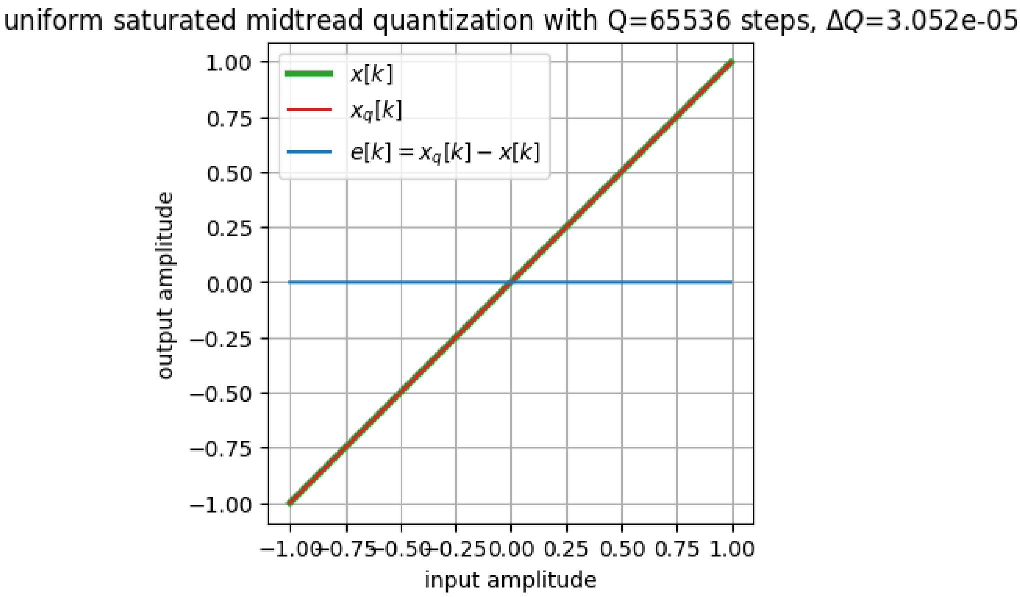
```python
# defining parameters
fs = 1800
N = 2*fs
k = np.arange (0, N)
fsin = 500
```

Picture 8. Implemented code

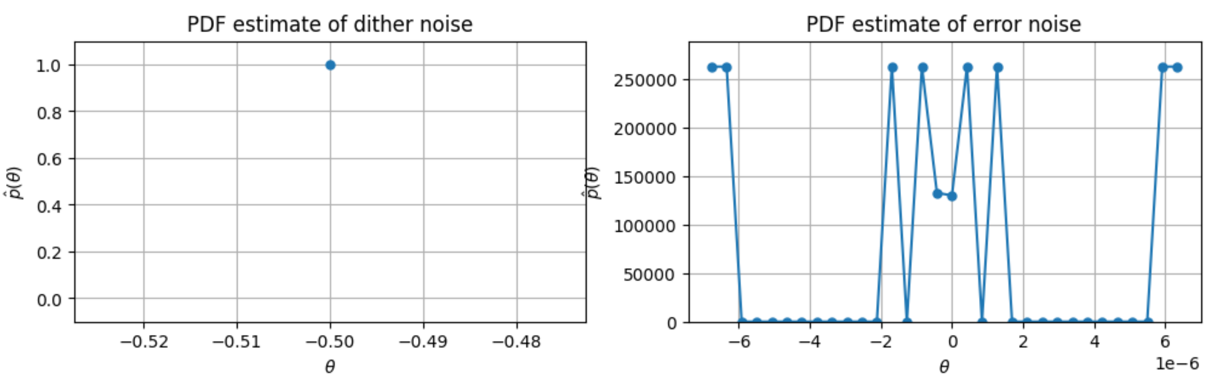
```python
# case 1
B = 16 # Bit
Q = 2**B # number of quanization steps
deltaQ = 1/(Q//2) # quantization step size
x = (1- deltaQ ) * np.sin(2 * np.pi * fsin/fs * k) # largest positive amplitude

plt.figure(figsize=(4,4))
check_my_quant(Q)
check_dithering(x=x,dither=x*0,Q=Q,case='case1')
```

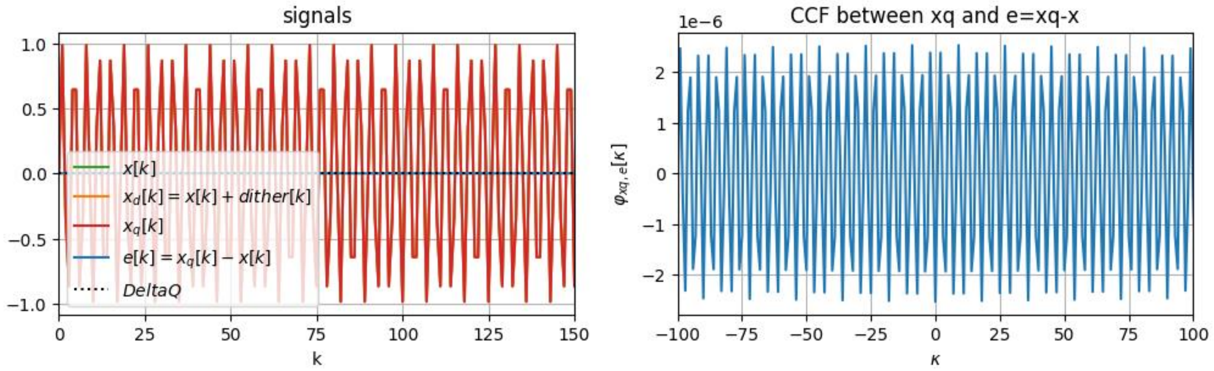
Picture 9. Implemented code



Picture 10. The result



Picture 11. The result

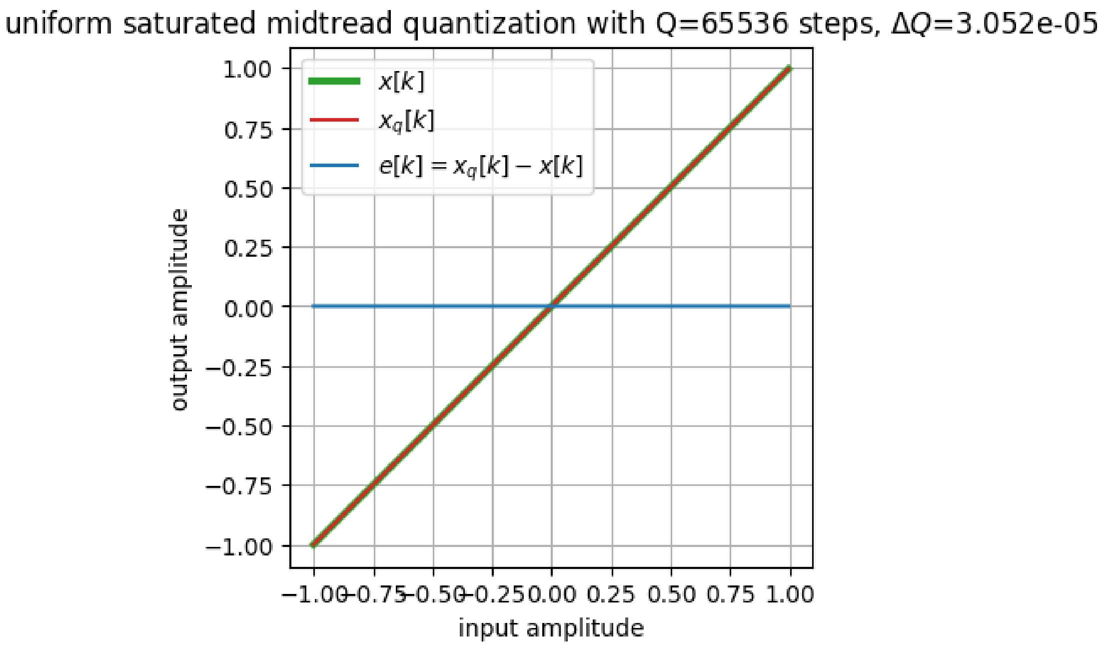


Picture 12. The result

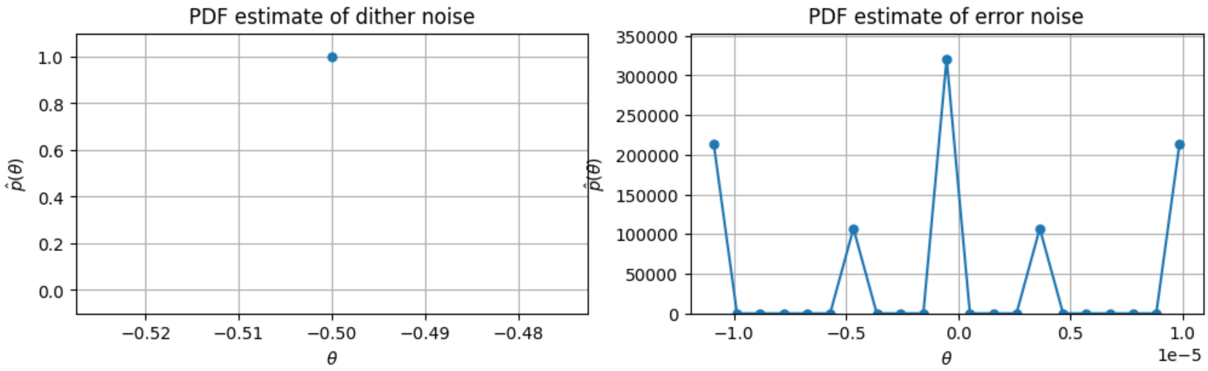
```
#case2
B=16
Q=2**B
deltaQ=1/(Q//2)
x=deltaQ*np.sin(2*np.pi*fsin/fs*k) #smallest amplitude

plt.figure(figsize=(4,4))
check_my_quant(Q)
check_dithering(x=x,dither=x*0,Q=Q,case='case2')
```

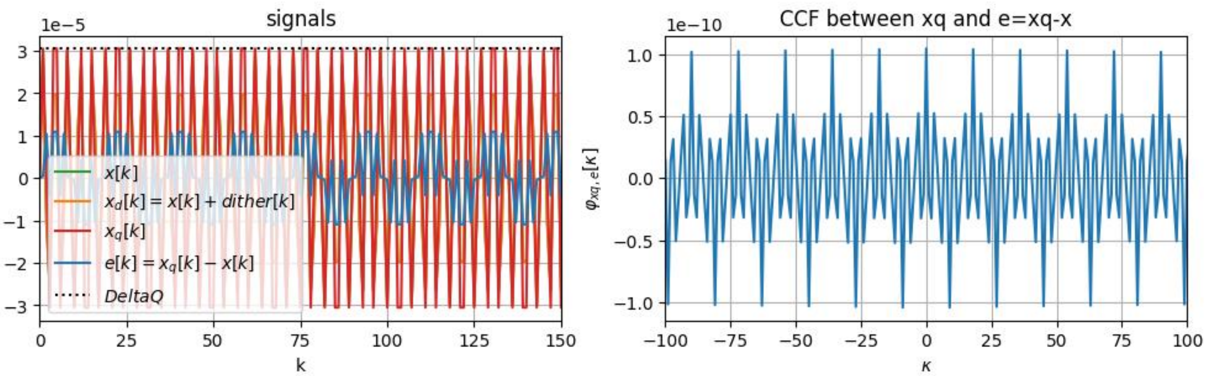
Picture 13. Implemented code



Picture 14. The result



Picture 15. The result

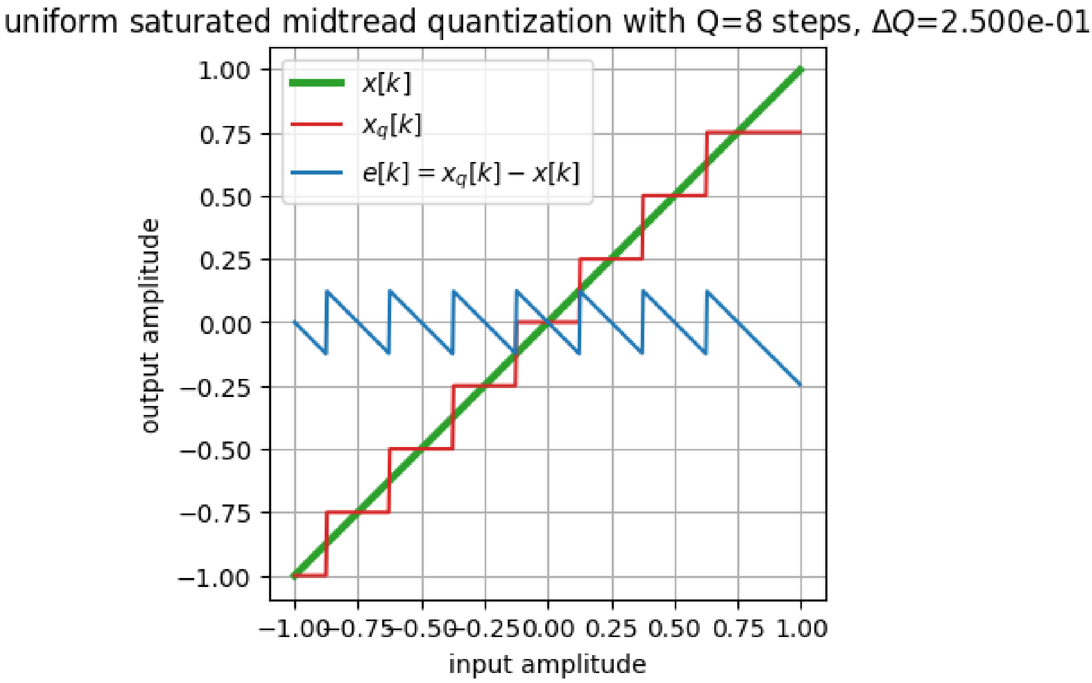


Picture 16. The result

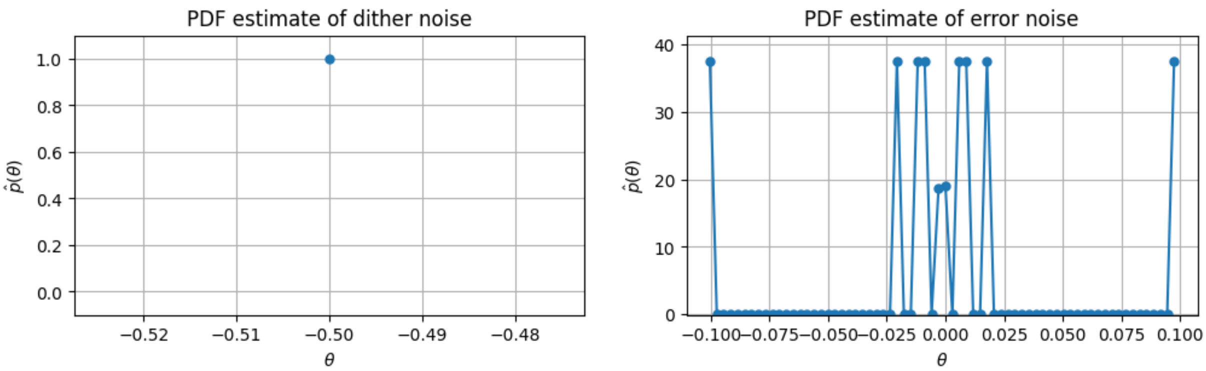
```
#case3
B=3
Q=2**B
deltaQ=1/(Q//2)
x=(1-deltaQ)*np.sin(2*np.pi*fsin/fs*k)

plt.figure(figsize=(4,4))
check_my_quant(Q)
check_dithering(x=x,dither=x*0,Q=Q,case='case3')
```

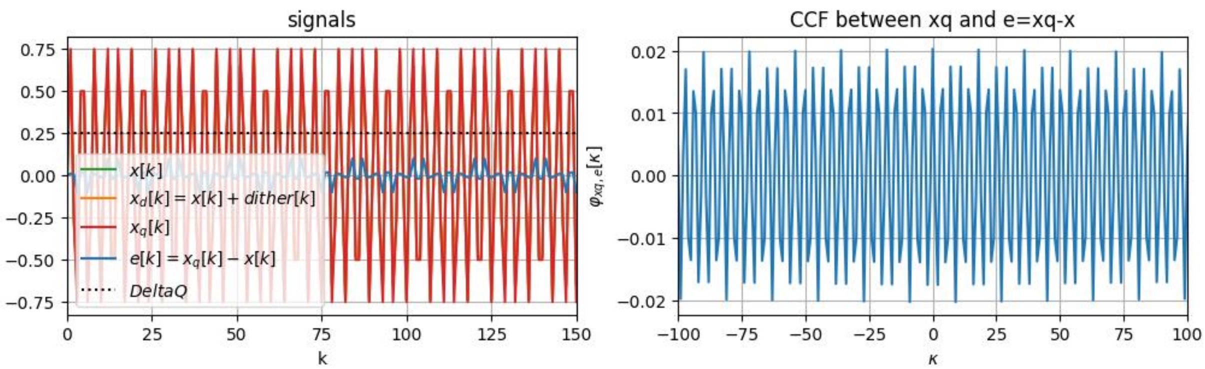
Picture 17. Implemented code



Picture 18. The result



Picture 19. The result

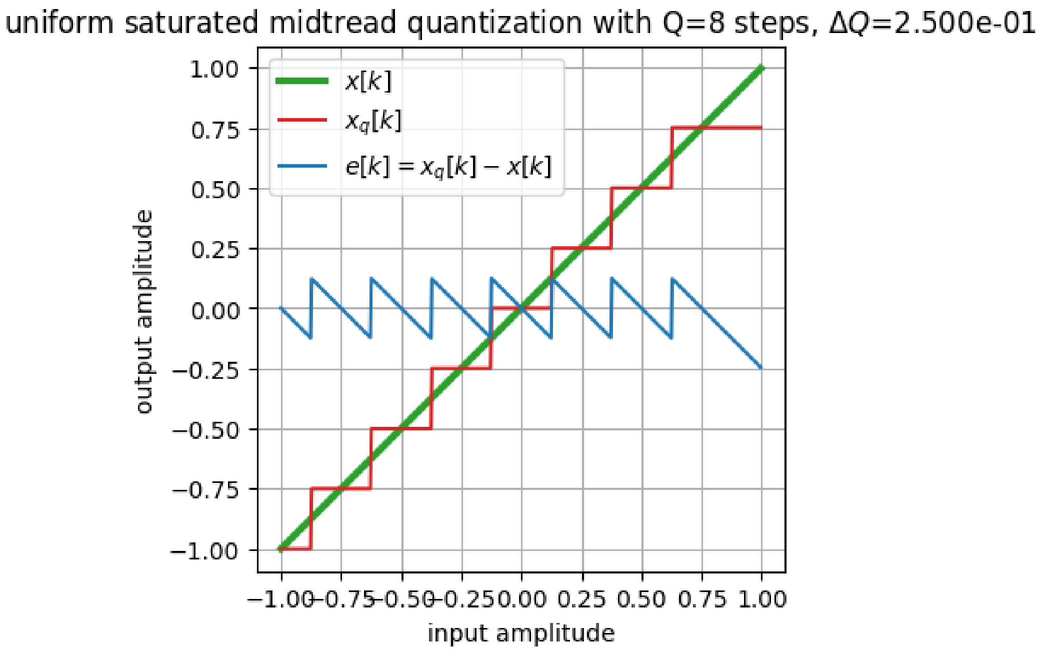


Picture 20. The result

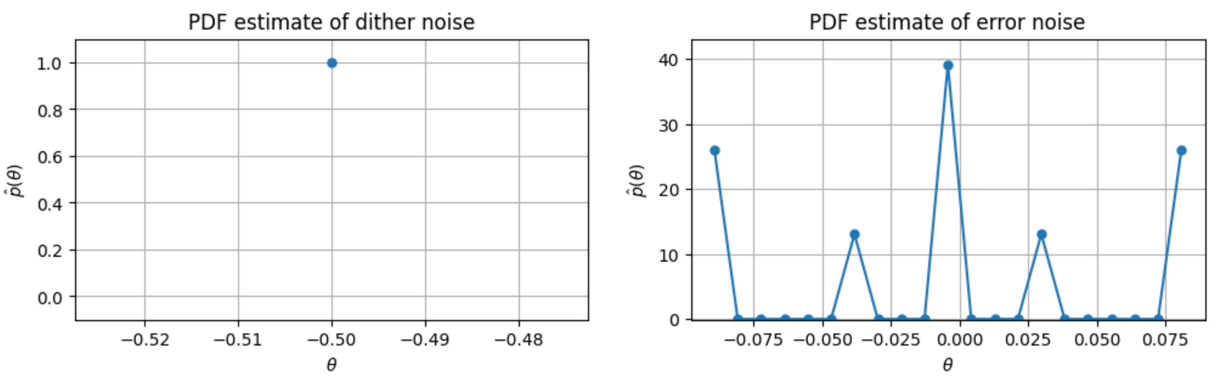
```
#case4
B=3
Q=2**B
deltaQ=1/(Q//2)
x=deltaQ*np.sin(2*np.pi*fsin/fs*k)

plt.figure(figsize=(4,4))
check_my_quant(Q)
check_dithering(x=x,dither=x*0,Q=Q,case='case4')
```

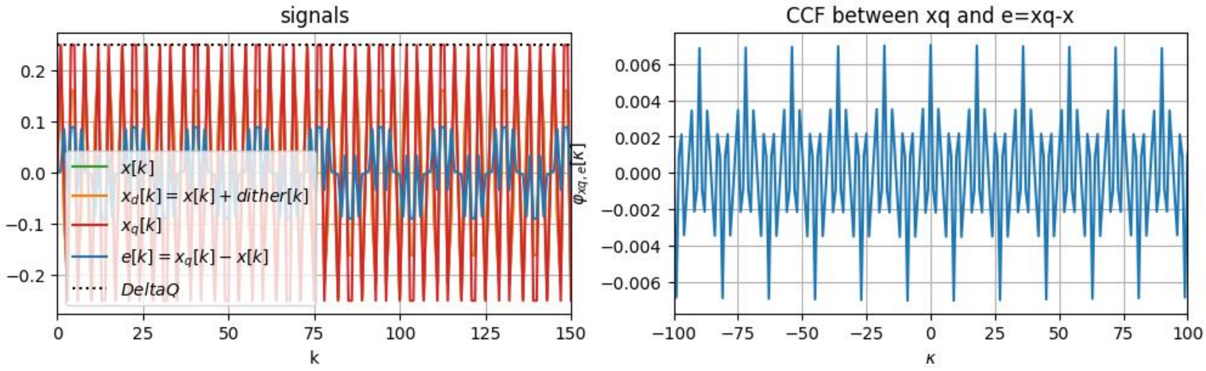
Picture 21. Implemented code



Picture 22. The result



Picture 23. The result

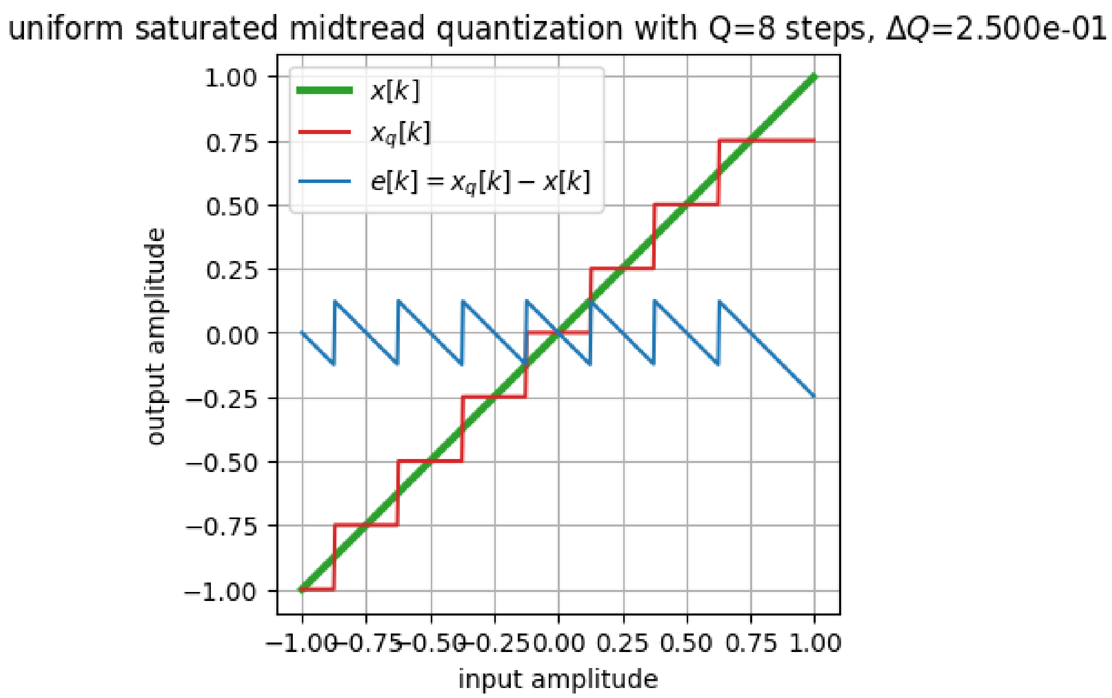


Picture 24. The result

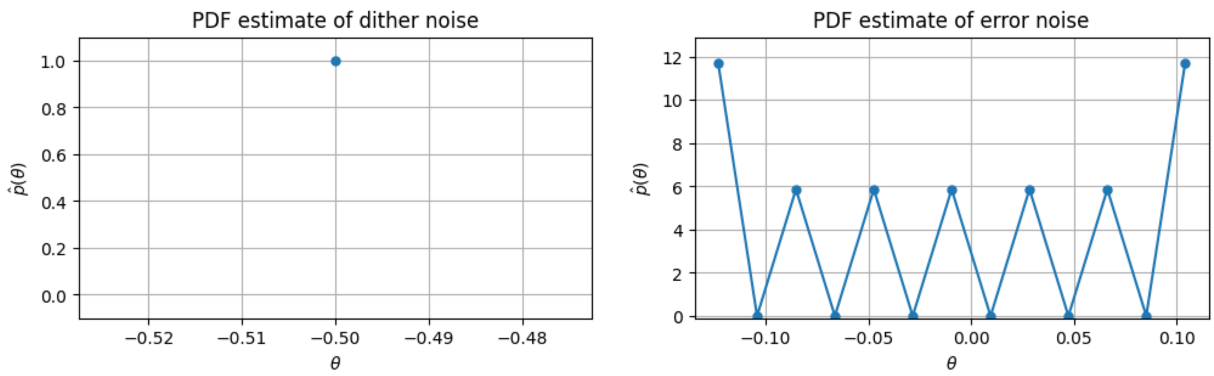
```
#case5
B=3
Q=2**B
deltaQ=1/(Q//2)
x=deltaQ/2*np.sin(2*np.pi*fsin/fs*k)

plt.figure(figsize=(4,4))
check_my_quant(Q)
check_dithering(x=x,dither=x*0,Q=Q,case='case5')
```

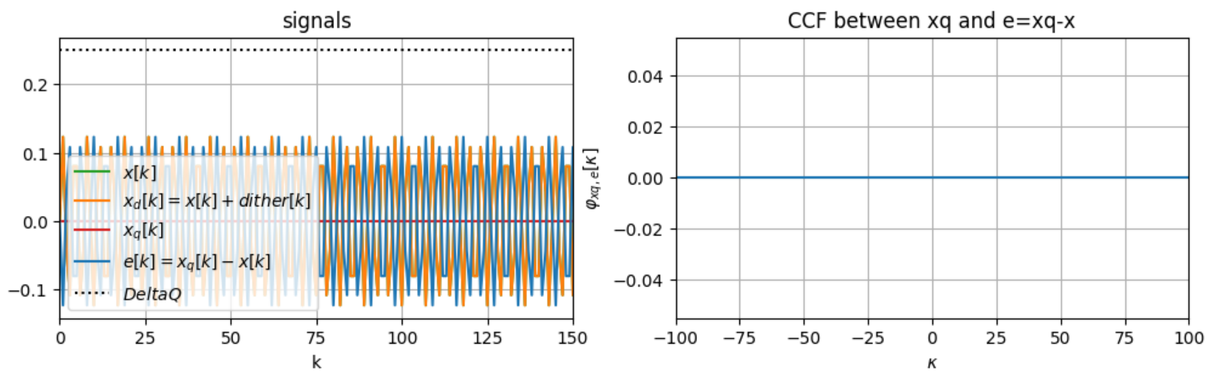
Picture 25. Implemented code



Picture 26. The result



Picture 27. The result



Picture 28. The result

5. Conclusions

This lab provided a comprehensive exploration of dithering and its impact on the quantization of signals. Through a series of detailed exercises, we gained practical experience in generating and applying dither noise, quantizing signals, and analyzing the effects of dithering.