Sprawozdanie z laboratorium Przetwarzanie Sygnałów i Obrazów

Ćwiczenie numer: 6

Temat: Filtry cyfrowe

Wykonujący ćwiczenie:

- Zaborowska Magda
- Wójtowicz Patryk

Studia dzienne I stopnia

Kierunek: Informatyka

Semestr: III Grupa zajęciowa: PS 12

Prowadzący ćwiczenie:
mgr inż. Dawid Najda

.....

OCENA

Data wykonania ćwiczenia

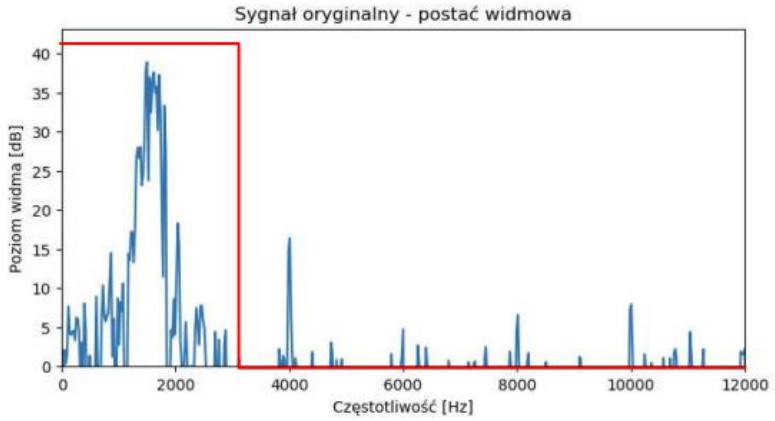
17.12.2021r.

.....

Data i podpis prowadzącego

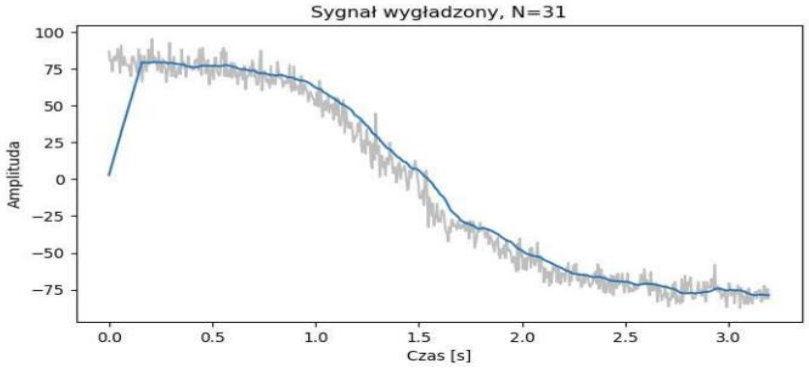
Teoria

Filtry cyfrowe to algorytmy przetwarzania sygnałów, po wykonaniu których z sygnału wejściowego usuwane są zbędne składowe. Można je zrealizować programowo albo sprzętowo. Najczęściej działają na dziedzinie częstotliwości tłumiąc pewien zakres częstotliwości i przepuszczając pozostałe częstotliwości. Jak widać na poniższym przykładzie z sygnału można odciąć niepotrzebna częstotliwości.



W przypadku gdy chcemy wygładzić sygnał, pozbyć się zaszumienia, każdą próbkę można zastąpić średnią z N ostatnich próbek sygnału.

$$y[n] = \frac{1}{N}\sum_{i=0}^{N-1} x[n-i]$$



Filtr cyfrowy posiada pamięć wewnętrzną, w której zapisywany jest stan, dzięki któremu odpowiedź na każdą kolejną próbkę nie zależy wyłącznie od tej próbki, ale również od innych próbek. W ten sposób może powstać filtr **przyczynowy**, jeśli zależność ta dotyczy wyłącznie próbek poprzednich lub **nieprzyczynowy**, jeśli uwzględniane są również próbki przyszłe.

Filtry cyfrowe można podzielić na dwie grupy:

- Filtry o skończonej odpowiedzi impulsowej (**SOI**, ang. finite impulse response – **FIR**)
- Filtry o nieskończonej odpowiedzi impulsowej (**NOI**, ang. infinite impulse response – **IIR**)

SOI (FIR):

Filtrem cyfrowym o skończonej odpowiedzi impulsowej nazywamy algorytm realizujący obliczenia wg tego równania $\boldsymbol{y[n] = \sum_{i=0}^{N-1} (\frac{1}{N}\, x[n-1])}$.

Filtr ten działa w następujący sposób:

- Bierze N ostatnich próbek,
- Mnoży je przez współczynnik 1/N,
- Sumuje wyniki mnożenia,
- Wysyła wynik na wyjście.

Odpowiedź impulsowa filtru FIR na pobudzenie impulsowe δ[n] jest równa zbiorowi współczynników filtru. Transmitancja jest transformatą Fouriera odpowiedzi impulsowej

NOI (IIR):

W technice analogowej działają one jak filtry FIR, filtrując niechciane częstotliwości. Transmitancja filtrów analogowych jest opisana w dziedzinie ciągłej zmiennej zespolonej s. Transmitancję w dziedzinie „analogowej" zmiennej s można przekształcić na transmitancję w dziedzinie „cyfrowej" zmiennej z za pomocą przekształcenia dwuliniowego:

$$s \to \frac{2}{T}\frac{z-1}{z+1} = \frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}$$

Transmitancja filtru drugiego rzędu przyjmuje postać:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

Metoda projektowania filtru cyfrowego:

- projektujemy filtr analogowy o zadanych parametrach,
- wykonujemy przekształcenie dwuliniowe,
- otrzymujemy transmitancję filtru cyfrowego.
- W odróżnieniu od filtrów FIR, te filtry mają współczynniki w liczniku i w mianowniku H(z).
- Filtry tego typu nazywamy filtrami cyfrowymi nieskończonej odpowiedzi impulsowej (IIR, infinite impulse filter).

Transmitancja filtru IIR:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_N z^{-N}}{1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_N z^{-N}}$$

- Wartość N oznacza rząd filtru (filter order).
- W filtrach IIR nie posługujemy się już pojęciem długości filtru.
- Filtr rzędu N ma N+1 współczynników b w liczniku i N współczynników a w mianowniku.
- Unormowana transmitancja ma zawsze a0 = 1.
- W algorytmach czasami podaje się a0, czasami się go pomija.

Zadanie 1

Treść zadania:

Napisać funkcję realizującą filtr cyfrowy opisany następującym równaniem różnicowym:
y[n] = b0x[n] + b1x[n − 1] + b2x[n − 2]–a1y[n − 1]–a2y[n − 2] dla parametrów ai, bj podanych przez prowadzącego.
Następnie:
• wygenerować odpowiedź impulsową filtru,
• zweryfikować poprawność implementacji porównując odpowiedź z podpunktu a) z odpowiedzią policzoną przy użyciu funkcji scipy.signal.lfilter

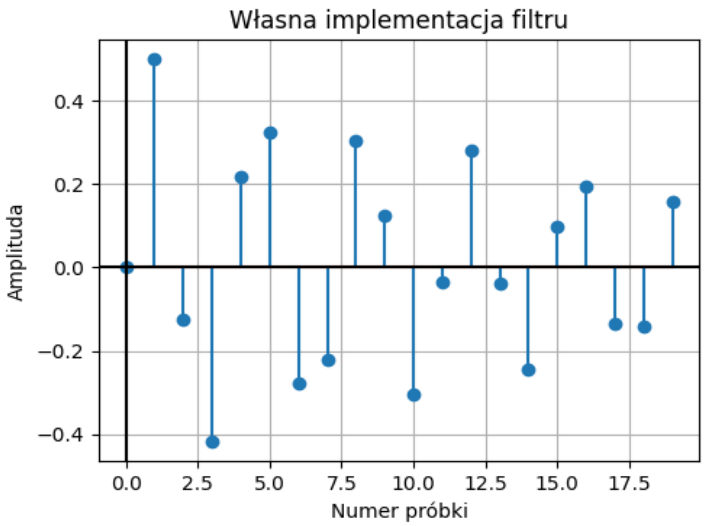
Realizacja w kodzie:

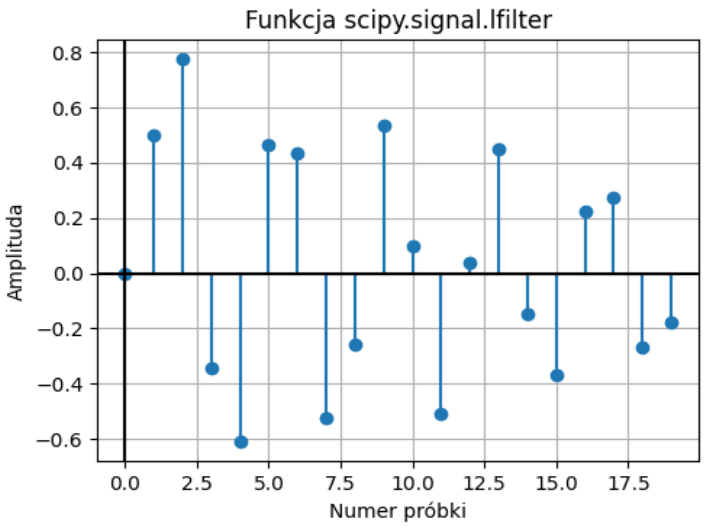
```python
from matplotlib import pyplot as plt
import numpy as np
from scipy import signal

def filter(x, a, b):
    y = []
    for i in range(0, len(x)):
        if(i == 0):
            val = b[0]*x[i]
            y.append(val)
        elif(i == 1):
            val = b[0]*x[i]+b[1]*x[i-1]-a[1]*y[i-1]
            y.append(val)
        else:
            val = b[0]*x[i]+b[1]*x[i-1]*b[2]*x[i-2]-a[1]*y[i-1]-a[2]*y[i-2]
            y.append(val)
    return y

a = [1, 0.25, 0.9]
b = [0.5, 0.9, 0.3]
x = np.zeros(20)
x[1] = 1
y = filter(x, a, b)
```

```python

plt.figure(figsize=(10,4),tight_layout=1)

plt.subplot(1,2,1)
plt.stem(y)
plt.title("Własna implementacja filtru")
plt.xlabel('Numer próbki')
plt.ylabel('Amplituda')
plt.grid(True,which='both')
plt.axvline(x=0,color='k')
plt.axhline(y=0,color='k')

y = signal.lfilter(b,a,x)
```

```python

plt.subplot(1,2,2)
plt.stem(y)
plt.title("Funkcja scipy.signal.lfilter")
plt.xlabel('Numer próbki')
plt.ylabel('Amplituda')
plt.grid(True,which='both')
plt.axvline(x=0,color='k')
plt.axhline(y=0,color='k')

plt.show()

```

Wyniki:





Zadanie 2

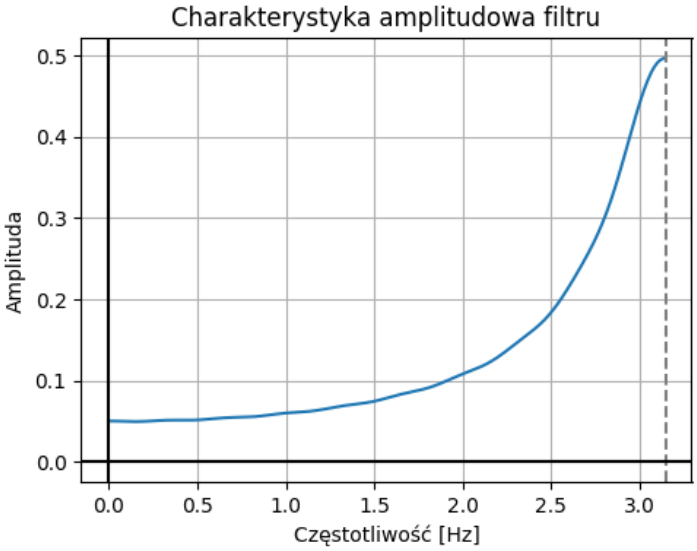
Treść zadania:

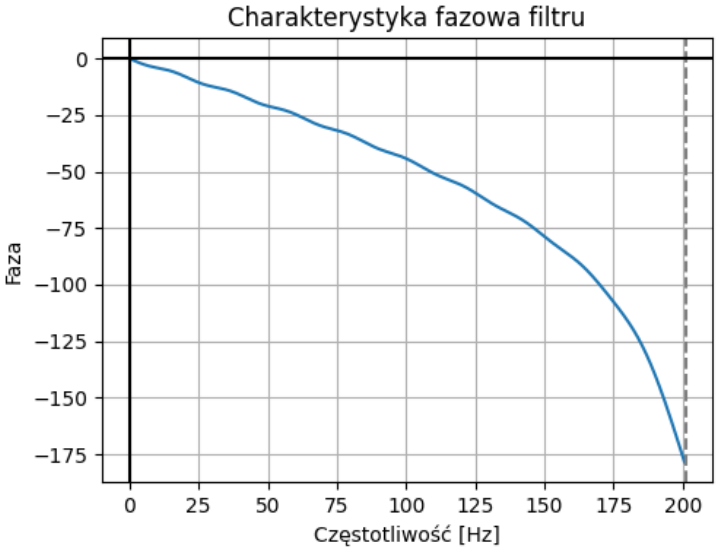
Wykorzystując funkcję scipy.signal.freqz obliczyć i wykreślić charakterystykę częstotliwościową (amplitudową oraz fazową) badanego filtru z zadania 1. Wykorzystując funkcję scipy.signal.tf2zpk wyznaczyć położenie zer i biegunów filtru z zadania 1 (przedstawić je na wykresie, na okręgu jednostkowym). • Co można powiedzieć o stabilności filtru?

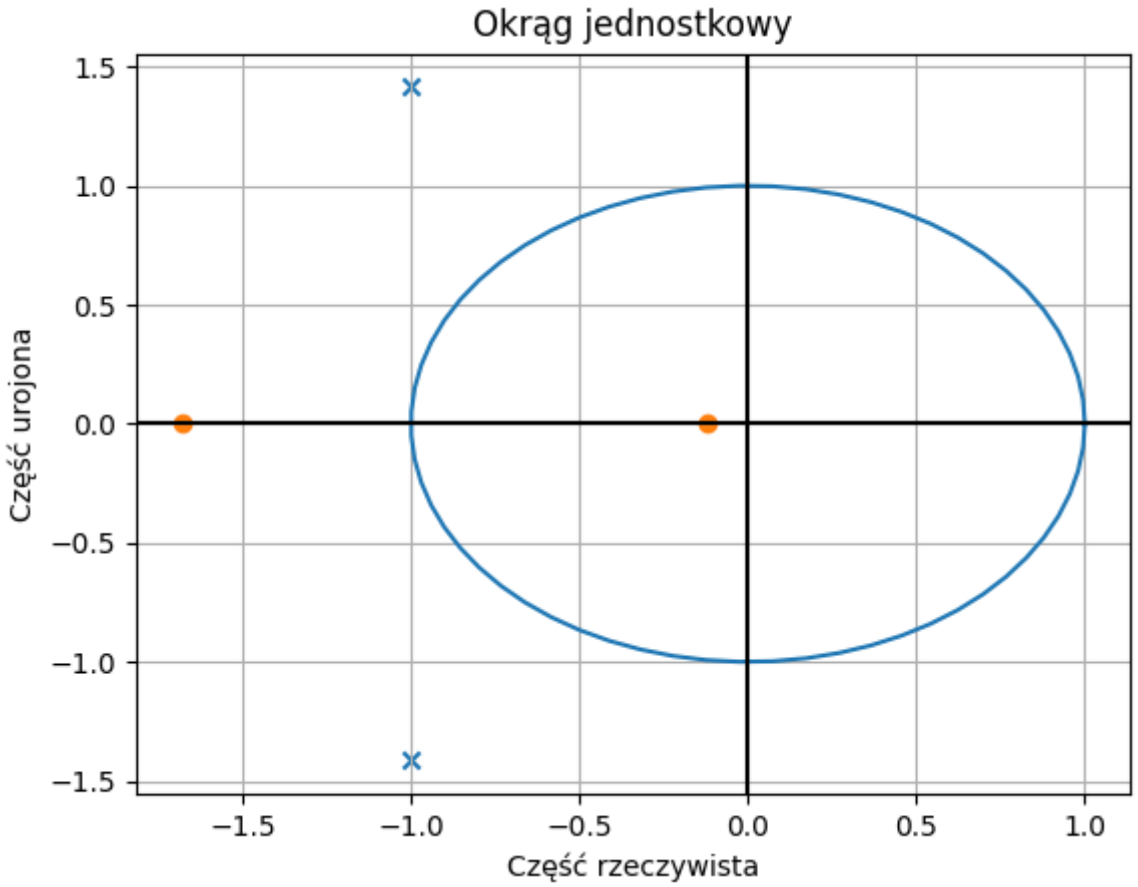
Realizacja w kodzie

```python
from matplotlib import pyplot as plt
import numpy as np
from scipy import signal

def filter(x, a, b):
    y = []
    for i in range(0, len(x)):
        if(i == 0):
            val = b[0]*x[i]
            y.append(val)
        elif(i == 1):
            val = b[0]*x[i]+b[1]*x[i-1]-a[1]*y[i-1]
            y.append(val)
        else:
            val = b[0]*x[i]+b[1]*x[i-1]*b[2]*x[i-2]-a[1]*y[i-1]-a[2]*y[i-2]
            y.append(val)
    return y

N = 64
a = [0.5, 0.9, 0.1]
b = [0.1, 0.2, 0.3]
x = np.zeros(20)
x[1] = 1
y = filter(x,a,b)
w, h = signal.freqz(y,1)

plt.figure(figsize=(10,4),tight_layout=1)
plt.subplot(1,2,1)
plt.plot(w,abs(h))
plt.title("Charakterystyka amplitudowa filtru")
plt.xlabel("Częstotliwość [Hz]")
plt.ylabel("Amplituda")
plt.grid(True,which='both')
plt.axvline(x=0,color='k')
plt.axvline(x=3.15,color='grey',linestyle='--')
plt.axhline(y=0,color='k')

f = (w*N)

plt.subplot(1,2,2)
plt.plot(f, np.degrees(np.unwrap(np.angle(h))))
plt.title("Charakterystyka fazowa filtru")
plt.xlabel("Częstotliwość [Hz]")
plt.ylabel("Faza")
plt.grid(True,which='both')
plt.axvline(x=0,color='k')
plt.axvline(x=201,color='grey',linestyle='--')
plt.axhline(y=0,color='k')

plt.show()

z, p, k = signal.tf2zpk(b, a)
s = np.linspace(0, 2*np.pi, 64)

plt.plot(np.cos(s), np.sin(s))
plt.scatter(np.real(z), np.imag(z), marker="x")
plt.scatter(np.real(p), np.imag(p), marker="o")
plt.title("Okrąg jednostkowy")
plt.xlabel('Część rzeczywista')
plt.ylabel('Część urojona')
plt.grid(True,which='both')
plt.axvline(x=0,color='k')
plt.axhline(y=0,color='k')

plt.show()
```

Wyniki







Zadanie 3

Treść zadania:

Zakładając, że sygnałem wejściowym x[n] jest szum gaussowski z parametrami μ = 0, σ2 = 1 (przynajmniej 64 próbki) wygenerować sygnał wyjściowy y[n], a następnie obliczyć i wykreślić widma amplitudowe sygnałów x[n] oraz y[n].

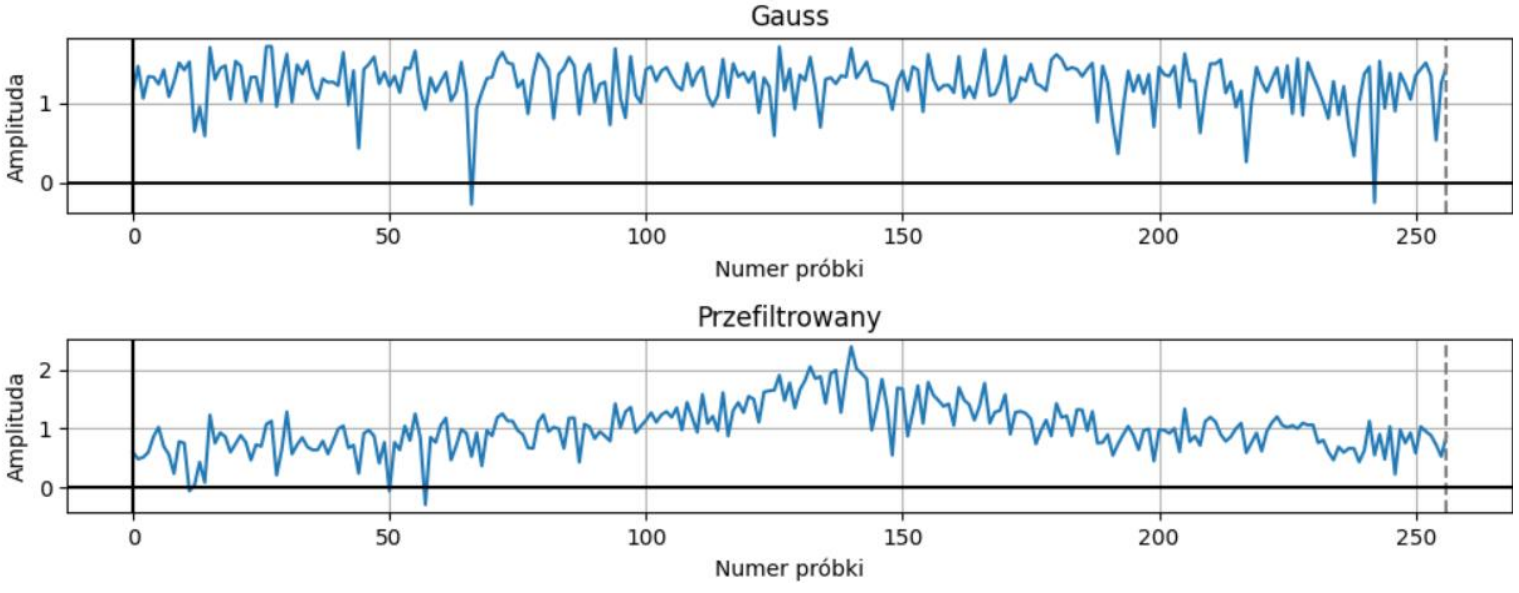
Realizacja w kodzie

```python
from matplotlib import pyplot as plt
import numpy as np

def filter(x, a, b):
    y = []
    for i in range(0, len(x)):
        if(i == 0):
            val = b[0]*x[i]
            y.append(val)
        elif(i == 1):
            val = b[0]*x[i]+b[1]*x[i-1]-a[1]*y[i-1]
            y.append(val)
        else:
            val = b[0]*x[i]+b[1]*x[i-1]*b[2]*x[i-2]-a[1]*y[i-1]-a[2]*y[i-2]
            y.append(val)
    return y

a = [1, 0.25, 0.9]
b = [0.5, 0.9, 0.3]
x = np.random.normal(0,1,512)
gauss = np.log10(np.abs(np.fft.rfft(x)))
fil= np.log10(np.abs(np.fft.rfft(filter(x,a,b))))
```

```python
plt.subplot(2,1,1)
plt.plot(gauss)
plt.title("Gauss")
plt.xlabel("Numer próbki")
plt.ylabel("Amplituda")
plt.grid(True,which='both')
plt.axvline(x=0,color='k')
plt.axvline(x=256,color='grey',linestyle='--')
plt.axhline(y=0,color='k')
```

```python
plt.subplot(2,1,2)
plt.plot(fil)
plt.title("Przefiltrowany")
plt.xlabel("Numer próbki")
plt.ylabel("Amplituda")
plt.grid(True,which='both')
plt.axvline(x=0,color='k')
plt.axvline(x=256,color='grey',linestyle='--')
plt.axhline(y=0,color='k')

plt.show()
```

Wyniki



Zadanie 4

Treść zadania:

Zaprojektować filtr cyfrowy do przetwarzania sygnału mowy. Przyjąć następujące założenia projektowe:

- filtr dolnoprzepustowy z nieskończoną odpowiedzią impulsową (IIR) o częstotliwości odcięcia równej 2 kHz,
- rząd filtru 3, rodzaj optymalizacji: Chebyshev (typ 1)
- Zafalowania w paśmie zaporowym nie powinny przekroczyć 1 dB
- Tłumienie w paśmie zaporowym powinno wynosić co najmniej 20 dB

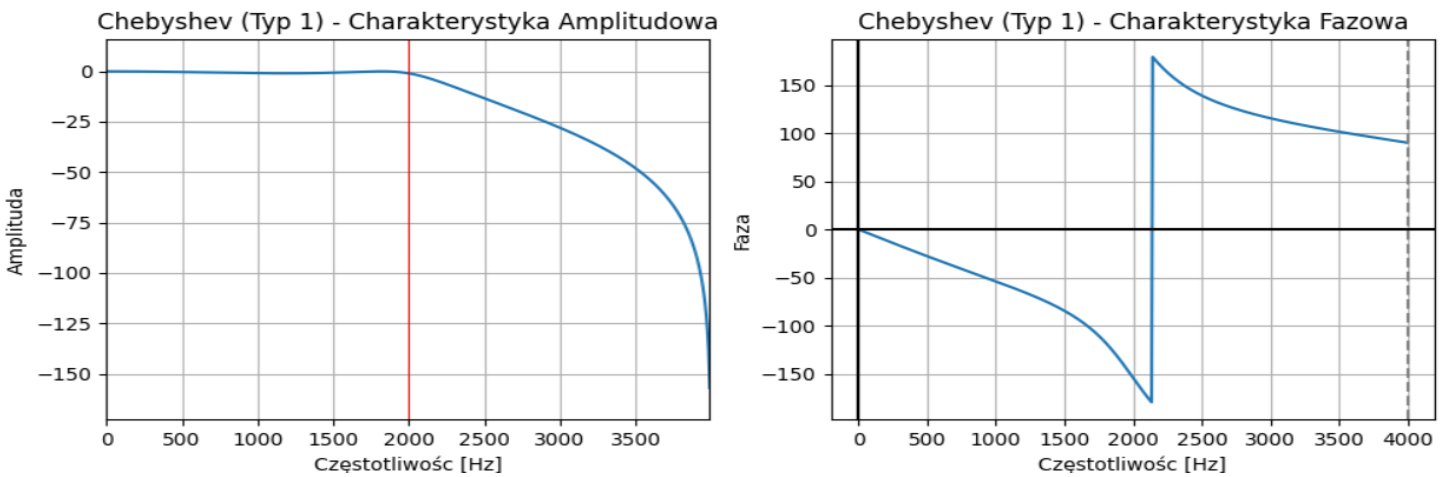
Należy przedstawić:

a) charakterystykę amplitudową i fazową zaprojektowanego filtru
b) krótki opis słowny słuchowej różnicy w brzmieniu sygnałów na wejściu i wyjściu filtru.

Realizacja w kodzie:

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal

fo = 2000
fp = 8000
fn = fp/2
rzad = 3
fal = 1
b, a = signal.cheby1(rzad, fal, fo/fn, btype='lowpass', analog = False)
w, h = signal.freqz(b, a)
f = fn*np.arange(len(w))/len(w)

plt.figure(figsize=(10,4),tight_layout=1)

plt.subplot(1,2,1)
plt.plot(f, 20 * np.log10(abs(h)))
plt.title('Chebyshev (Typ 1) - Charakterystyka Amplitudowa')
plt.xlabel('Częstotliwośc [Hz]')
plt.ylabel('Amplituda')
plt.margins(0, 0.1)
plt.axvline(fo, color='r', linewidth=0.75)
plt.grid(True,which='both')
plt.axvline(x=0,color='k')


plt.subplot(1,2,2)
plt.plot(f,np.degrees(np.angle(h)))
plt.title('Chebyshev (Typ 1) - Charakterystyka Fazowa')
plt.xlabel('Częstotliwośc [Hz]')
plt.ylabel('Faza')
plt.grid(True,which='both')
plt.axvline(x=0,color='k')
plt.axvline(x=4000,color='grey',linestyle='--')
plt.axhline(y=0,color='k')

plt.show()
```

Wyniki:



Zadanie 10 – dla ambitnych

Treść zadania:

W celu stworzenia obrazu binarnego zaimplementować algorytm binaryzacji Niblacka. Algorytm Niblacka korzysta z okna o ustalonym rozmiarze w celu obliczenia średniej wartości pikseli oraz odchylenia standardowego. Po obliczeniu średniej m oraz odchylenia standardowego s, obliczana jest wartość r zgodnie ze wzorem r = ts + m, gdzie t = 0.8. Po obliczeniu wartości r dany piksel jest ustawiany na czarny lub biały w zależności od tego, czy bieżący r jest większy od bieżącego piksela. Wartość bieżącego piksela można obliczyć, biorąc pod uwagę średnią z trzech kanałów, wartość maksymalną kanału, minimalną lub w inny sposób.

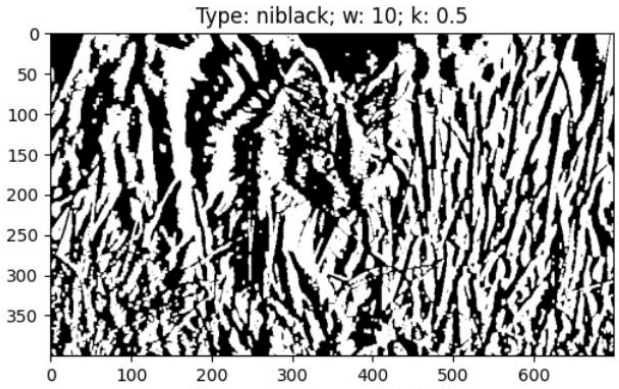
- Sprawdzić i przeanalizować wyniki binaryzacji Niblacka dla różnych t oraz różnego rozmiaru okien.
- Zaimplementować binaryzację Sauvola. Binaryzacja Sauvola od Niblacka różni się użytym wzorem: $r = m \cdot (1 + t \cdot (s/d - 1))$, gdzie t = 0.5, d = 2
- Sprawdzić i przeanalizować wyniki binaryzacji Sauvola dla różnych t, d oraz różnego rozmiaru okien.

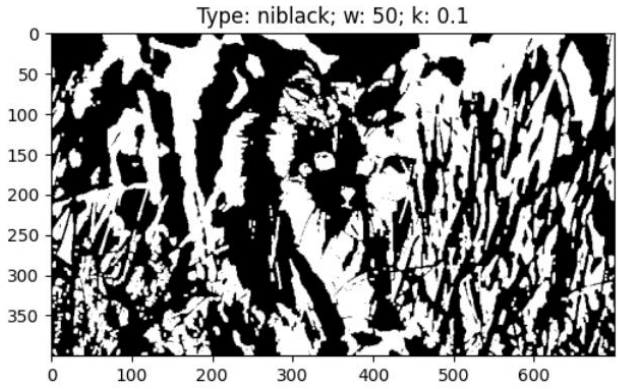
Realizacja w kodzie:

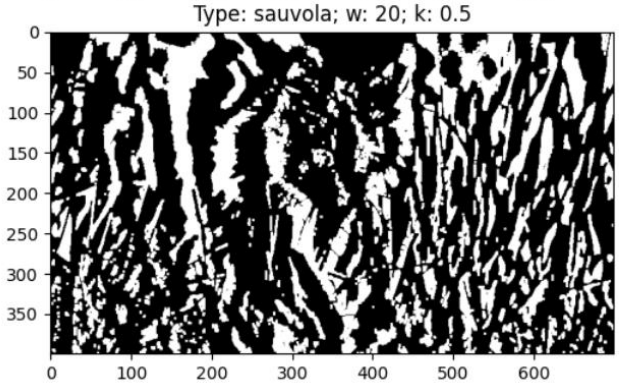
```
25  photo = cv2.imread('kot.jpg')
26  plt.imshow(photo)
27
28  KERNEL = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))
29  RESIZE = (700,400)
30
31  photo = cv2.imread('kot.jpg')
32  plt.imshow(photo)
33
34  KERNEL = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))
35  RESIZE = (700,400)
36
37  bin_photos = mb.binarize(photo, bin_methods, resize=RESIZE, morph_kernel=KERNEL, return_original=True)
38  original_photo = bin_photos[0]
39
40  fig=plt.figure(figsize=(10,4),tight_layout=1)
41
42  plot_idx = 1
43
44  for idx, bin_photo in enumerate(bin_photos[1:]):
45      title = "Type: {}; w: {}; k: {}".format(bin_methods[idx]['type'], bin_methods[idx]['window_size'], bin_methods[idx]['k_factor'])
46      fig.add_subplot(2, 2, plot_idx, title=title)
47      plt.imshow(bin_photo, cmap='gray')
48      plot_idx += 1
49  plt.show()
```

```
51  candidate_boxes = []
52  candidate_centroids = []
53
54  for bin_photo in bin_photos[1:]:
55      plot_photo = original_photo.copy()
56      boxes, centroids = ut.get_candidate_regions(bin_photo)
57      candidate_boxes += boxes
58      candidate_centroids += centroids
59      plot_photo =ut.plot_blobs(plot_photo, boxes)
60      fig.add_subplot(2, 2, plot_idx)
61      plt.imshow(plot_photo, cmap='gray')
62      plot_idx += 1
63
64  plt.show()
65
66  fig=plt.figure(figsize=(10,4),tight_layout=1)
67
68  plot_photo = original_photo.copy()
69
70  ut.plot_blobs(plot_photo, candidate_boxes, plot_line=True, centroids=candidate_centroids)
71  plt.imshow(plot_photo)
```

Obraz źródłowy i wynik:







Interpretacja wyników i wnioski

Filtry cyfrowe charakteryzuje szereg zalet. Jedną z nich jest łatwość uzyskania charakterystyk częstotliwościowych o w zasadzie dowolnym kształcie w pasmie przepustowym, dużym tłumieniu w pasmie zaporowym i wąskim pasmie przejściowym, które są trudno osiągalne lub nawet niemożliwe do uzyskania w wypadku filtrów analogowych. Ponadto charakterystykę filtrów cyfrowych można łatwo zmienić przez modyfikację programu.

W zadaniu pierwszym tworzono własną funkcję filtru o skończonej odpowiedzi impulsowej. W porównaniu z gotową funkcją bibliotekową widać delikatną różnicę.

W zadaniu kolejnym należało obliczyć i wykreślić charakterystykę częstotliwościową i wyznaczyć położenie zer i biegunów filtru z zadania 1. Filtr okazał się być niestabilnym. Można to rozpoznać po położeniu biegunów na wykresie okręgu jednostkowego. Jeśli bieguny oznaczone „x" znajdują się na okręgu lub wewnątrz niego, filtr jest stabilny. W przypadku gdy znajdują się poza okręgiem, filtr jest niestabilny.

W zadaniu trzecim widać przefiltrowany sygnał mocno przypomina charakterystykę częstotliwościową filtra. W zadaniu kolejnym, po odsłuchu sygnału mowy, zauważono, że po użyciu filtru Chebyshev wysokie częstotliwości zostały usunięte.

W zadaniu 10 wykonywano binaryzację Sauvola i Niblacka. Binaryzacja jest procesem konwersji obrazów kolorowych lub monochromatycznych do obrazu dwupoziomowego (binarnego). Celem binaryzacji jest redukcja informacji zbędnej i pozostawienie informacji istotnej z punktu widzenia dalszego przetwarzania. Stanowi najprostszą metodę segmentacji obrazu, czyli podziału obrazu na rozłączne regiony charakteryzujące się jednorodnością wartości pikseli lub innej branej pod uwagę cechy. W binaryzacji Niblacka Próg ustalany jest na podstawie średniej oraz odchylenia standardowego wartości pikseli z jego otoczenia. Metoda progowania Sauvola i Pietikainena jest uważana obecnie za jedną z najlepszych metod binaryzacji obrazów. W binaryzacji Savola wartość progu ustalana jest na podstawie średniej oraz odchylenia standardowego wartości pikseli w bloku bezpośrednio otaczającym analizowany piksel. W obszarze dużego kontrastu metoda zachowuje się bardzo podobnie do metody Niblacka. Różnica pojawia się dla obszarów o małym kontraście, dla których następuje wyraźnie obniżenie progu w stosunku do średniej.

Źródła

[1] https://elektronikab2b.pl/technika/33352-projektowanie-filtrow-cyfrowych
[2] https://multimed.org/student/pdio/pdio06b_filtry_cyfrowe_IIR.pdf
[3] https://pl.wikipedia.org/wiki/Filtr_cyfrowy
[4] https://livesound.pl/tutoriale/4876-filtry-cyfrowe-w-praktyce.-filtry-fir
[5] https://multimed.org/student/pdio/pdio06a_filtry_cyfrowe_FIR.pdf