Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie

AGH University of Science
and Technology

Sonar kierunkowy oparty o technikę beamformingu

Miłosz Derżko, Piotr Książek

Projekt z przedmiotu Zaawansowane Techniki Przetwarzania Sygnałów

Część I

Wstęp

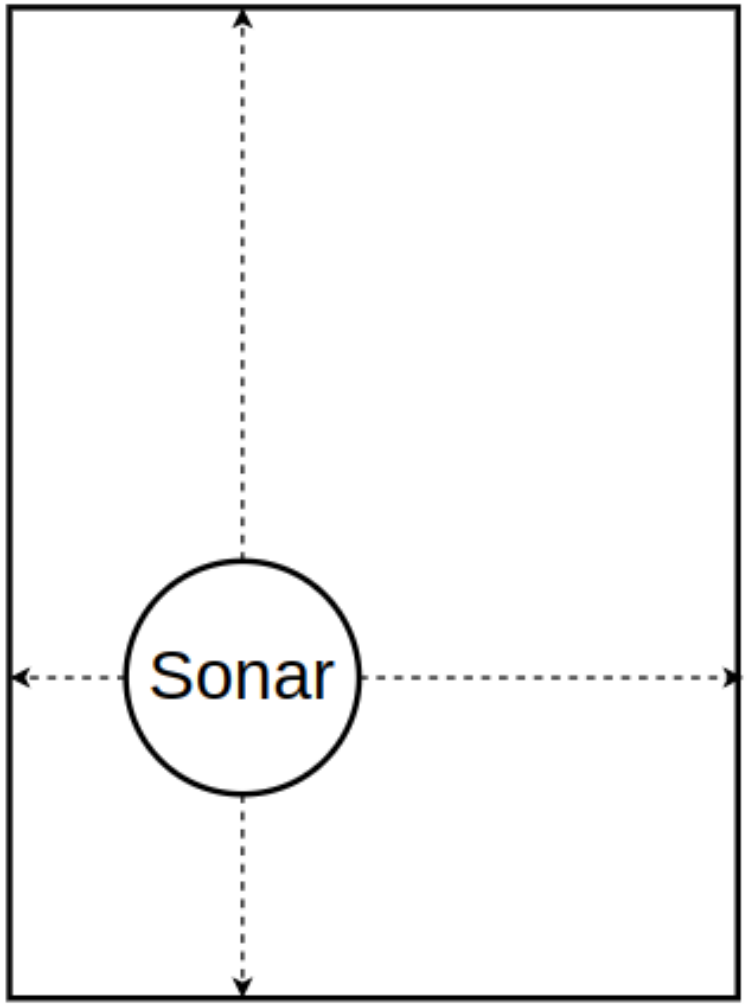
Założenia

- Symulacja pokoju (pyroomacoustics)
- Przestrzeń 2D
- Pomieszczenie prostokątne
- Źródło wszechkierunkowe
- Beamformer 4 kanałowy

Budowa sonaru

Rysunek: Sonar kierunkowy

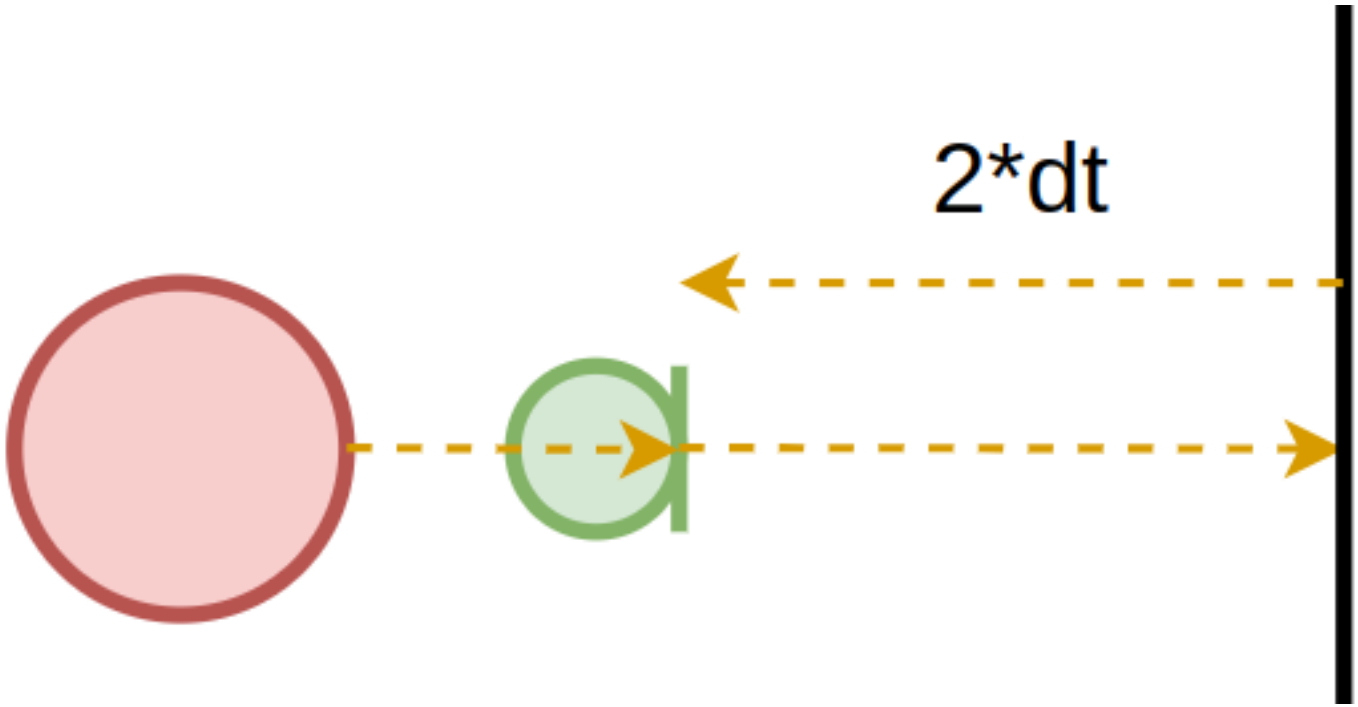
Położenie



Działanie

- Odtwarzamy sygnał dźwiękowy
- Nasłuchujemy odpowiedzi z mikrofonów
- Dokonujemy analizy

Działanie

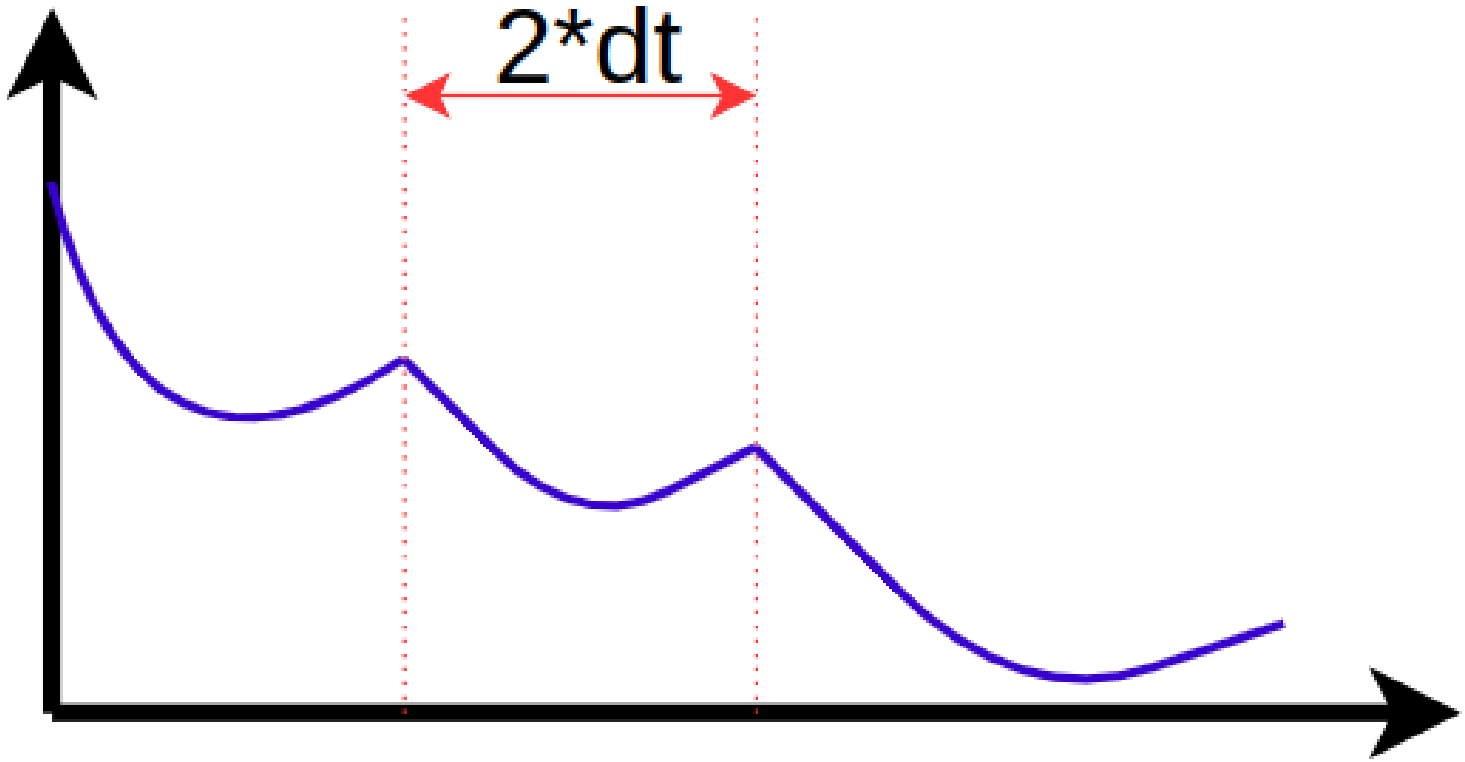


Rysunek: Działanie sonaru

Analiza

- Tworzymy cztery beamformery dla każdego kierunku
- Wyznaczamy sygnały dla odpowiednich wektorów własnych
- Wyznaczamy autokorelacje dla każdego kierunku

Oczekiwany rezultat



Rysunek: Spodziewana funkcja autokorelacji dla jednego z kanałów

AGH

www.agh.edu.pl

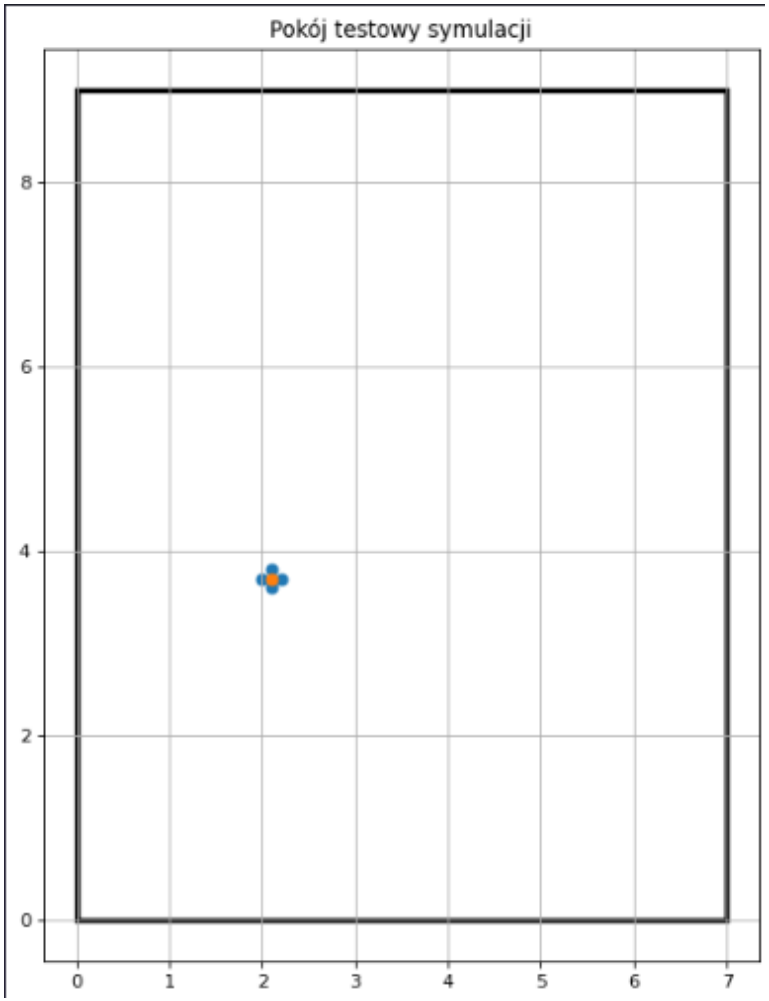
Część II

Model pokoju

Model pokoju

- Python
- Pyroomacoustics
- Wymiary pokoju 7m x 9m
- Pozycja sonaru [2.1m, 3.7m]

Wizualizacja pokoju



Część III

Pyroomacoustics

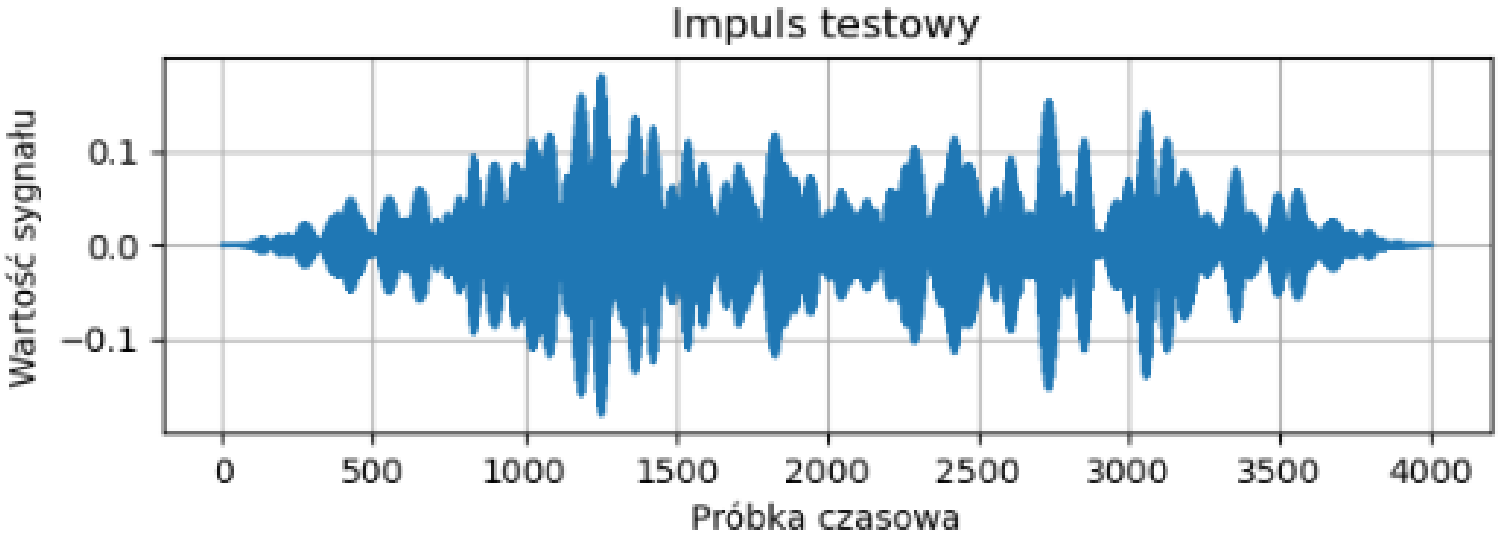
Biblioteka

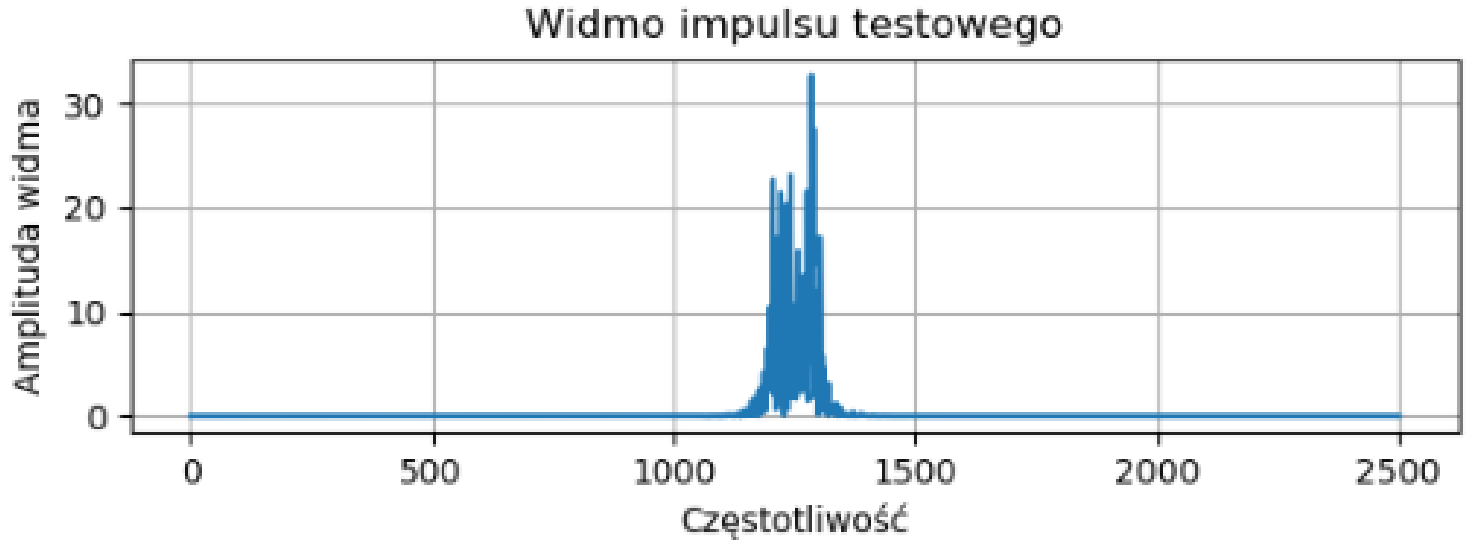
- Dokumentacja na stronie `https://github.com/LCAV/pyroomacoustics`
- Używa metody geometrycznej

Stworzenie sygnału testowego


```
fs = 5000
imp_length = 0.8 # w sekundach

test_signal = np.random.random(int(fs*imp_length))-0.5

b, a = butter(5, [1200, 1300], fs=fs, btype='band')
test_signal = lfilter(b, a, test_signal)

test_window = tukey(int(fs*imp_length))
test_signal = test_window*test_signal

test_spectrum = rfft(test_signal)
df = fs/test_signal.shape[0]
fvector = np.arange(0,test_spectrum.shape[0])*df
```

Sygnał testowy





Stworzenie Modelu pokoju

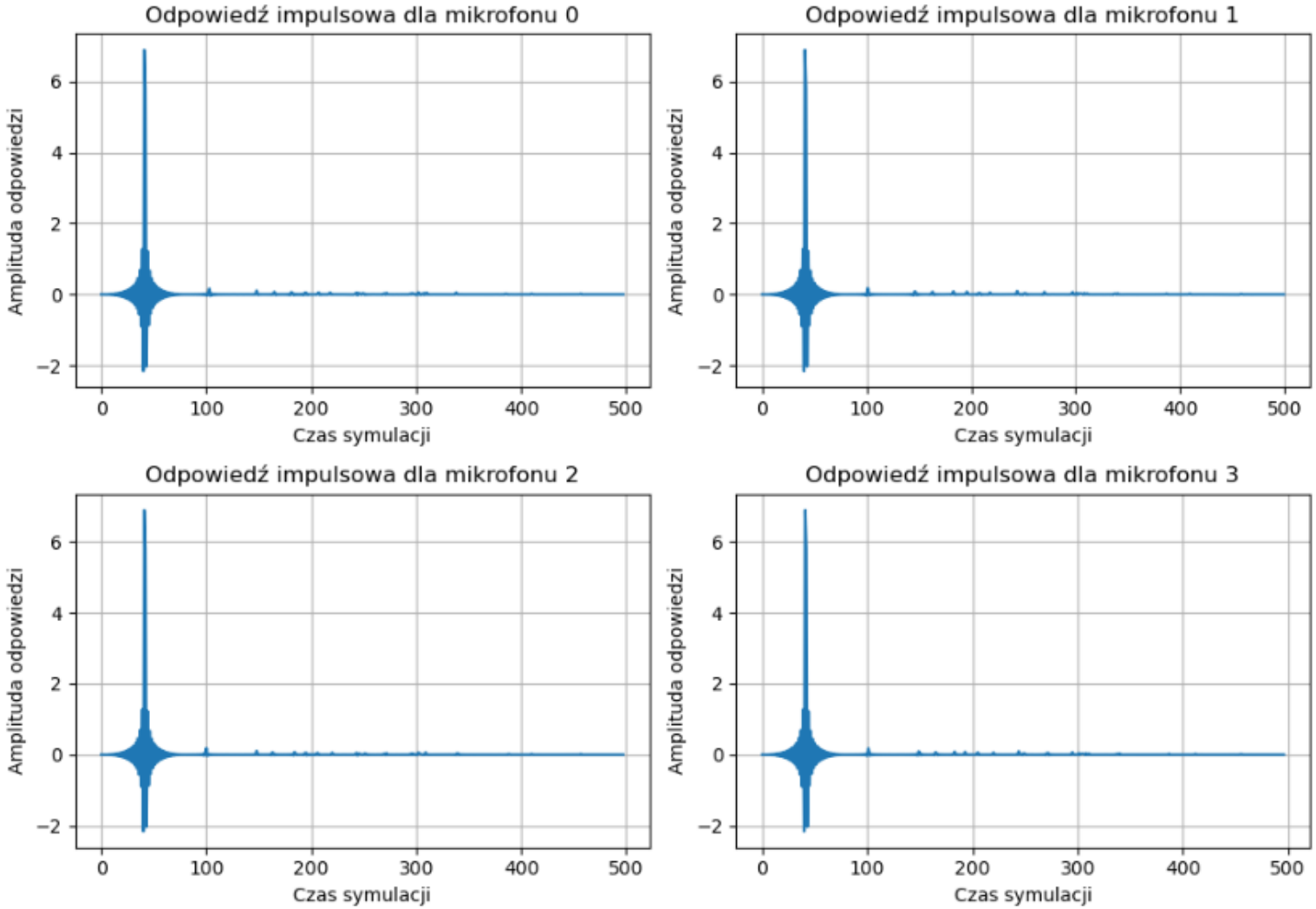

```
room_dimensions = [7,9]
src_position = [2.1, 3.7]
mic_distance = 0.1 # przykładowa wartość
mic_locs = np.c_[
        [src_position[0]+mic_distance, src_position[1]],
        [src_position[0], src_position[1]-mic_distance],
        [src_position[0]-mic_distance, src_position[1]],
        [src_position[0], src_position[1]+mic_distance],
]

material = pra.Material(0.1)
test_room = pra.ShoeBox(
                    room_dimensions,
                    fs=fs,
                    materials=material,
                    max_order=3
                    )
test_room.add_source(src_position, signal=test_signal)
test_room.add_microphone_array(mic_locs)
test_room.compute_rir()
test_room.simulate()
```

Stworzenie Modelu pokoju

```
plt.figure(figsize=(10, 7))
for i in range(4):
        plt.subplot(2,2,i+1)
        plt.plot(test_room.rir[i][0])
        plt.title("Odpowiedź impulsowa dla mikrofonu %d"%i)
        plt.xlabel("Czas symulacji")
        plt.ylabel("Amplituda odpowiedzi")
        plt.grid(True, "both")

plt.tight_layout()
```

www.agh.edu.pl

Odpowiedzi impulsowe z każdego mikrofonu



Część IV

Wykonanie

Założenia implementacyjne

- Wykonanie założeń projektu
- Stosowanie gotowych rozwiązań tam gdzie to możliwe (podejście pythonowe)
- Implementacja proof-of-concept

Problemy implementacji

- Trudność spełnienia założeń projektu
- Gotowe rozwiązania niedostatecznie udokumentowane lub wprost niedziałające
- Implementacja ręczna zbyt kosztowna czasowo

Zmiana założeń wdrożeniowych

W związku z napotkanymi problemami, pewne parametry wykonawcze zostały zmodyfikowane. Przede wszystkim zmieniona została liczba mikrofonów w macierzy oraz pasmo sygnału testowego.

Sygnał testowy


```
imp_length = 0.8 # w sekundach
test_signal = np.random.random(int(fs*imp_length))-0.5
b, a = butter(5, [700, 800], fs=fs, btype='band')
test_signal = lfilter(b, a, test_signal)
test_window = tukey(int(fs*imp_length))
test_signal = test_window*test_signal
test_spectrum = rfft(test_signal)
df = fs/test_signal.shape[0]
fvector = np.arange(0,test_spectrum.shape[0])*df
```

Pomieszczenie testowe

```
Lg_t = 0.100
Lg = np.ceil(Lg_t*fs)
room_bf = pra.ShoeBox([20,40], fs=fs, max_order=12)
source = np.array([10, 20])
room_bf.add_source(source, delay=0., signal=test_signal)
center = [10, 20]; radius = 0.1
fft_len = 512
echo = pra.circular_2D_array(center=center, M=12, phi0=0,
radius=radius)
mics = pra.Beamformer(echo, room_bf.fs, N=fft_len, Lg=Lg)
room_bf.add_microphone_array(mics)
```

Obliczenia PyRoomAcoustics

AGH

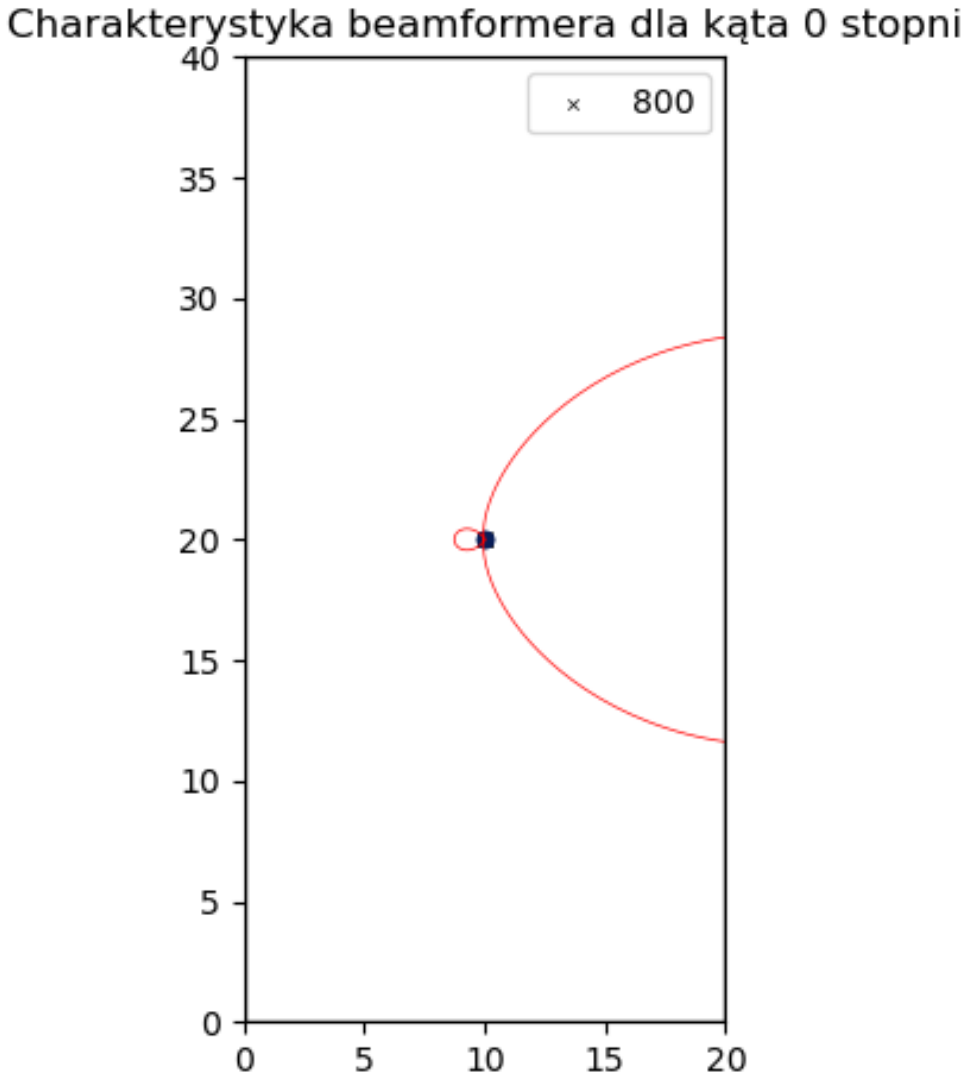
```
mics.far_field_weights(np.deg2rad(angle))
room_bf.compute_rir()
room_bf.simulate()

signal_das = mics.process(FD=False)
sd.play(signal_das/10, fs)

fig, ax = room_bf.plot(freq=[800], img_order=0)
ax.legend(['800'])
plt.title(f"Charakterystyka beamformera dla kąta {angle} stopni")
plt.savefig(f"img\\signal_{angle}_room_beamformer.png")
```

www.agh.edu.pl

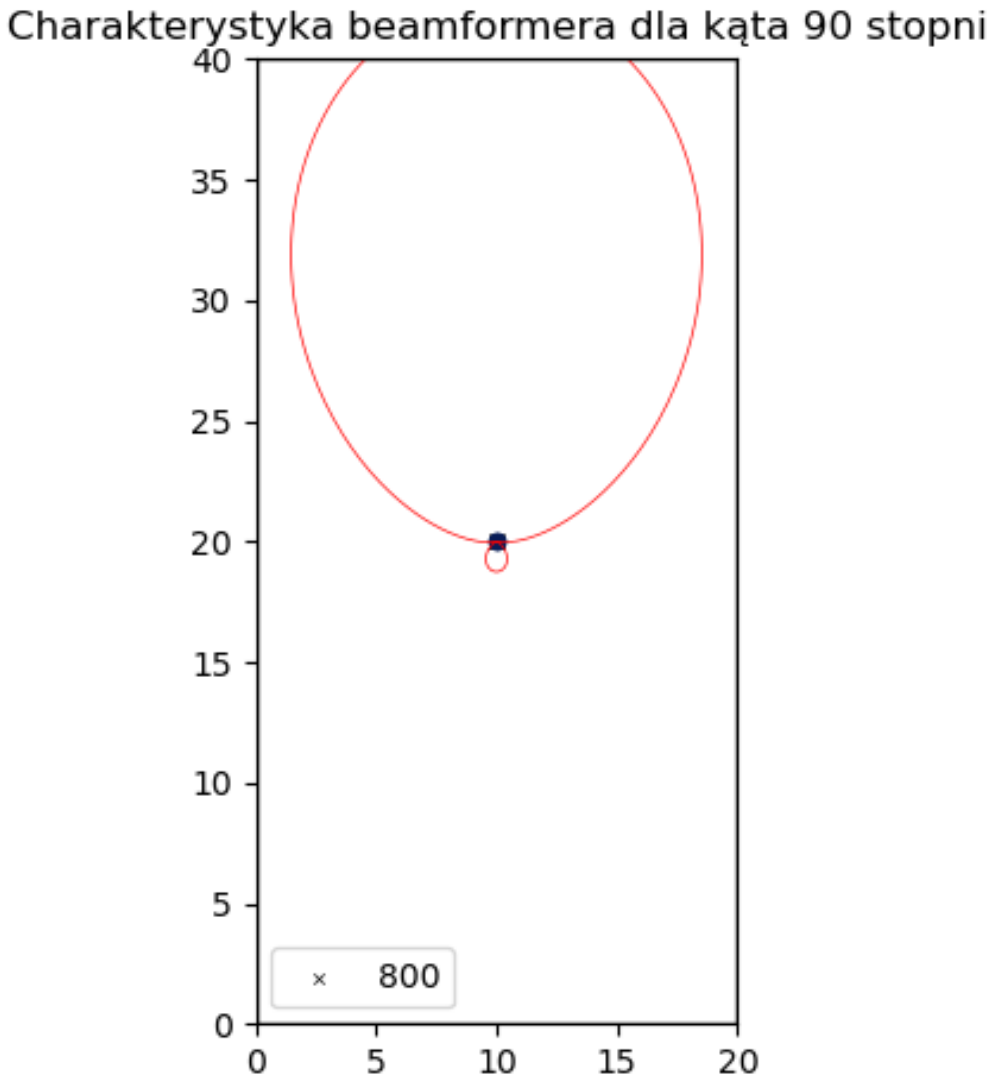
Wyznaczanie opóźnień

```
corr = np.correlate(signal_das, signal_das, mode="full")
corr_oneside = np.abs(corr[int(np.shape(corr)[0]/2):])
peak_indices, peak_dict = scipy.signal.find_peaks(corr_oneside,
distance = 100, height=(None, None))
peak_heights = peak_dict['peak_heights']
highest_peak_index = peak_indices[np.argmax(peak_heights)]
second_highest_peak_index =
peak_indices[np.argmax(np.delete(peak_heights,
np.argmax(peak_heights)))+1]
reflection_delay = ...
(second_highest_peak_index - highest_peak_index)/fs
reflection_distance = 343*reflection_delay/2
print(reflection_distance)
```

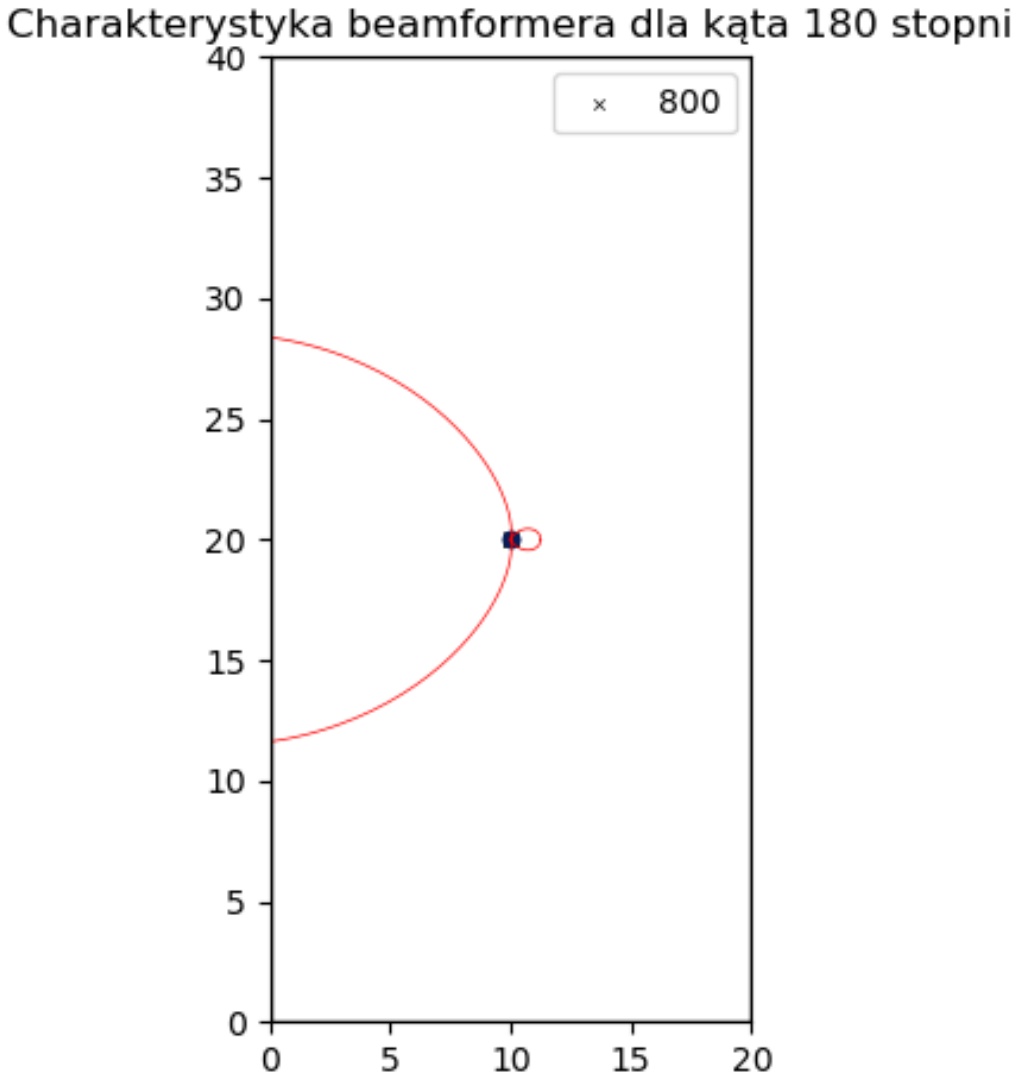
Beamformer dla każdego z kątów



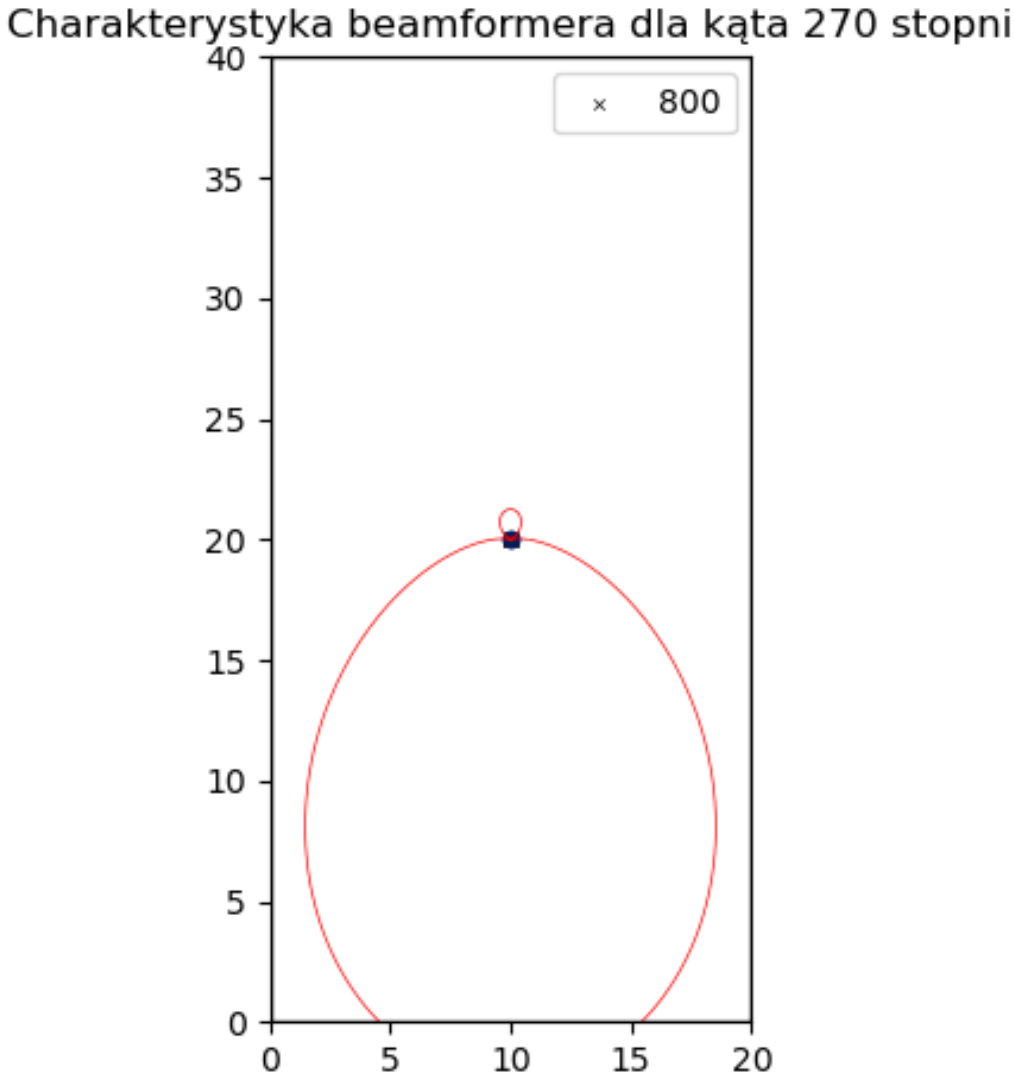
Beamformer dla każdego z kątów



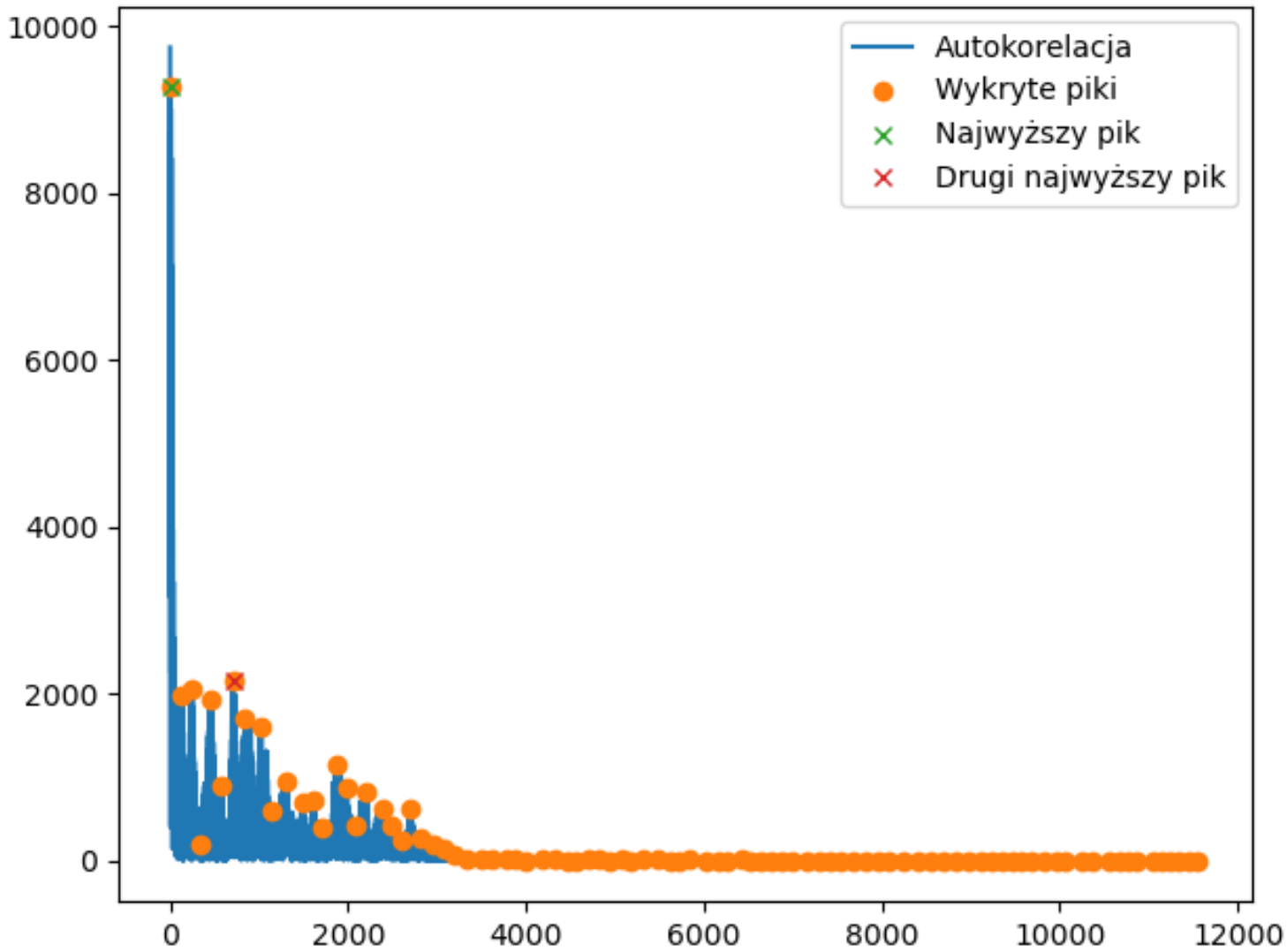
Beamformer dla każdego z kątów



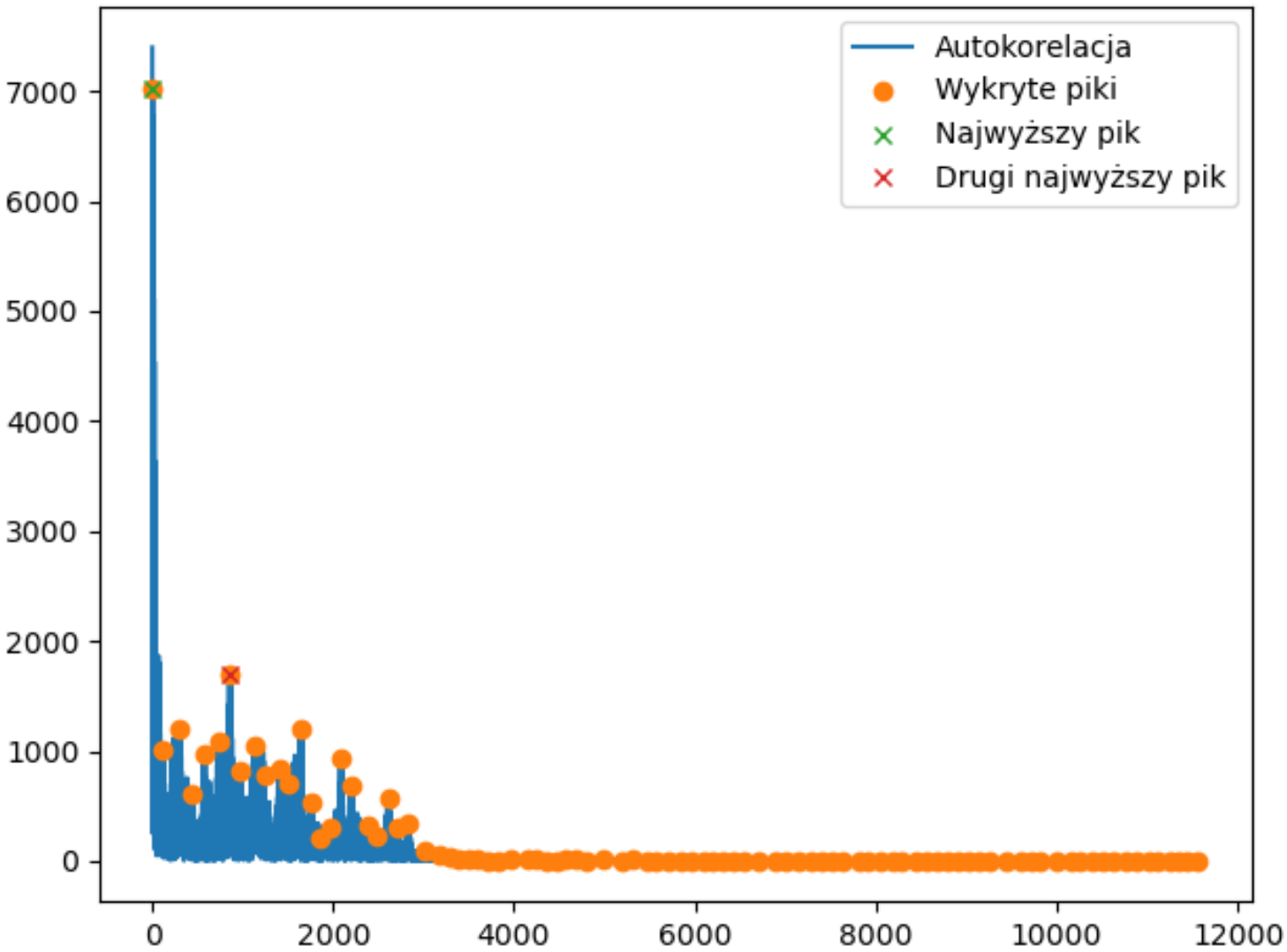
Beamformer dla każdego z kątów



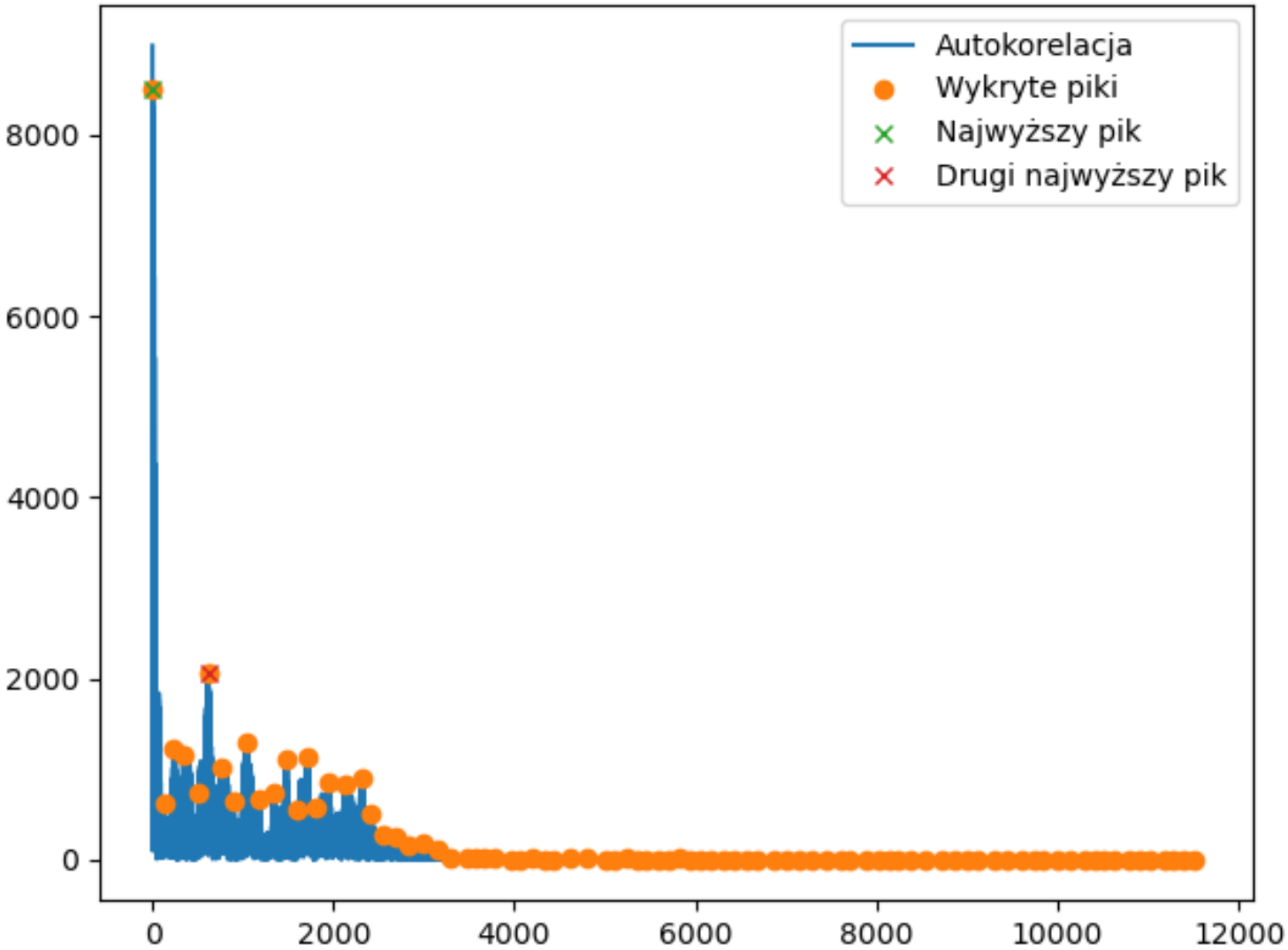
Autokorelacja dla każdego z kątów



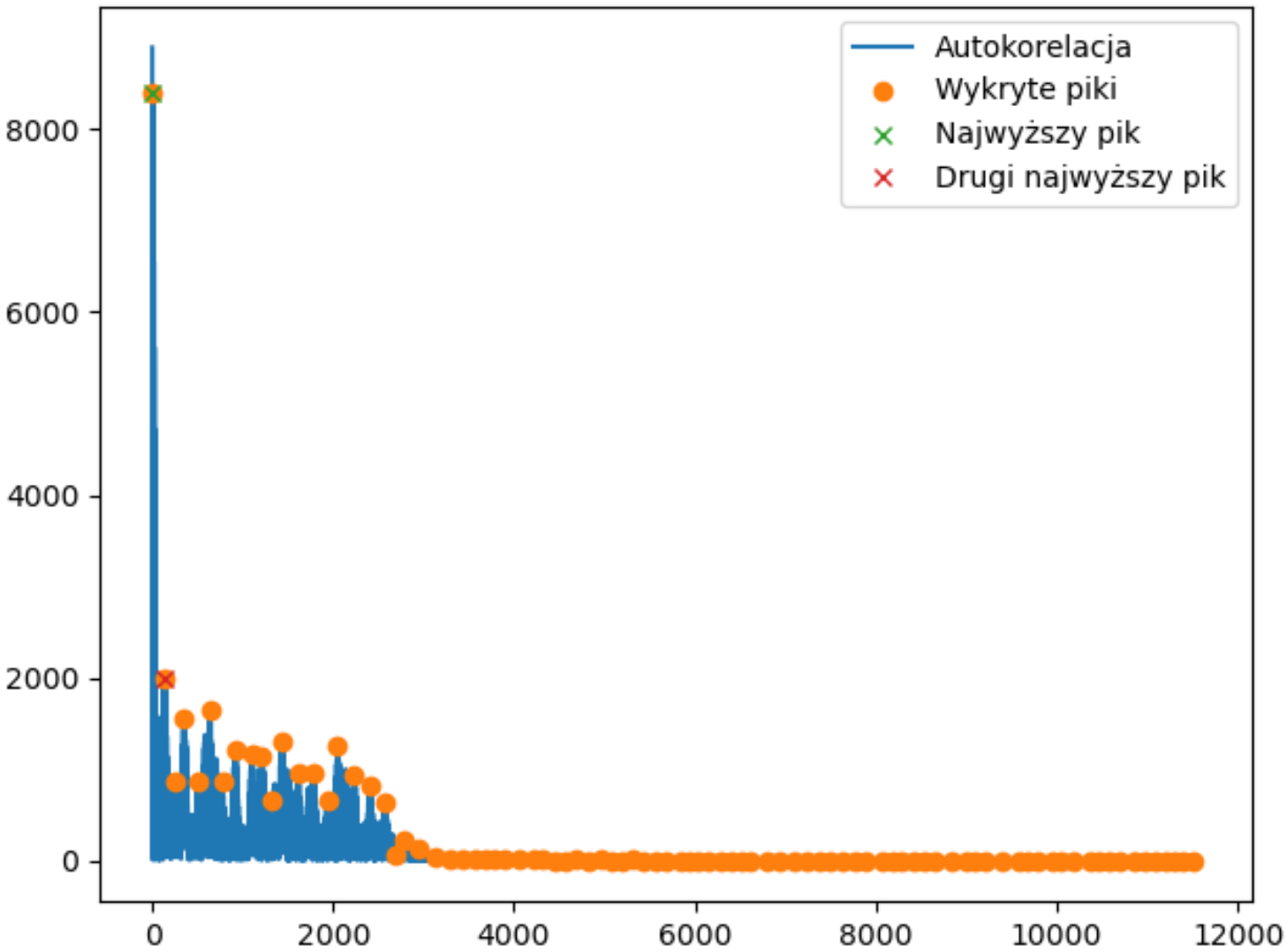
Autokorelacja dla każdego z kątów



Autokorelacja dla każdego z kątów



Autokorelacja dla każdego z kątów



Wyniki działania metody

- Kąt 0, Rzeczywisty:10 m Wynik: 9,4 m
- Kąt 90, Rzeczywisty:20 m Wynik: 18,0 m
- Kąt 180, Rzeczywisty:10 m Wynik: 3,9 m
- Kąt 270, Rzeczywisty:20 m Wynik: 9,1 m

Losowość wyników

Wyniki uzyskiwane przez metodę wahają się o nawet kilka metrów. W rzeczywistej implementacji można rozważyć zastosowanie wielokrotnego testowania i uśredniania wyników.

Działanie metody

Metoda nie oferuje satysfakcjonujących wyników. Wymaga ona dalszego dopracowania.

Możliwości rozwoju

- Zastosowanie innych algorytmów kształtowania wiązki
- Zastosowanie innych potencjalnych geometrii macierzy mikrofonowej
- Pogłębienie przeglądu literatury - odnalezienie rozwiązania do weryfikacji
- Rozważenie zmiany zakresu częstotliwości na znacznie wyższy, analogicznie wobec dalmierzy ultradźwiękowych.