УДК 004.045:621.396.967.2(043.2)

Труханська В. О., здобувачка вищої освіти,
Ніколюк П. К., д-р фіз.-мат. наук,
професор, професор кафедри
інформаційних технологій

ОБРОБКА СИГНАЛІВ ДЛЯ ВИЯВЛЕННЯ ПОВІТРЯНИХ ЦІЛЕЙ

Донецький національний університет імені Василя Стуса, м. Вінниця

Виявлення повітряних цілей, як-от дрони, літаки або ракети, є одним з актуальних завдань сучасної радіолокації та акустичного моніторингу. У міру розвитку технологій зростає потреба у використанні автоматизованих систем для ідентифікації цілей, що базуються на аналізі акустичних сигналів. Такі системи дають змогу виконувати моніторинг у реальному часі, що знижує ризики невиявлення небезпечних об’єктів [1]. Завдяки використанню методів цифрової обробки сигналів (ЦОС) завдання виявлення цілей може бути вирішене з високою точністю навіть у складних умовах, зокрема за наявності шуму або перешкод.

У цій роботі представлено методологію аналізу акустичних сигналів для виявлення характерних частот, пов’язаних з повітряними цілями. Реалізований алгоритм базується на швидкому перетворенні Фур’є (FFT), методах пошуку піків і порівнянні виявлених частот із наперед заданим набором цільових характеристик [2].

Процес виявлення повітряних цілей акустичними методами включає кілька ключових етапів. Спершу йде попередня обробка сигналу, де здійснюється завантаження аудіофайла та приведення сигналу до моноформату. Далі здійснюється спектральний аналіз: за допомогою швидкого перетворення Фур’є (FFT) отримується частотний спектр сигналу, що допомагає перейти від часової області до частотної [1]. Оскільки більшість інформаційних частот містяться у позитивному спектрі, аналіз обмежується цим діапазоном. Результатом є амплітудний спектр сигналу. Наступне – це виявлення характерних частот, використовується алгоритм пошуку піків для ідентифікації частот із високою амплітудою. Ці частоти порівнюються з наперед заданим списком характерних частот, що відповідають потенційним цілям (наприклад, дронам, літакам тощо) [3]. Алгоритм допускає певний поріг амплітуди для уникнення помилкових виявлень через шум. Останнім кроком є візуалізація результатів, що представляються у вигляді графіків, які демонструють сигнал у часовій області та його спектр у частотній. На спектрі також вказуються виявлені частоти цілей.

Розроблений алгоритм реалізовано мовою Python із використанням бібліотек numpy, matplotlib та scipy. Код представляє процес обробки сигналу, попередньо завантаживши аудіофайл, і графічний вивід представлення результатів.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import wavfile
from scipy.signal import find_peaks
# === Параметри для виявлення цілей ===
```

```
# Частоти, характерні для повітряних цілей (дрони, літаки, ракети)
target_frequencies = [150, 300, 450, 800, 1200]  # Цілі (можна розширити)
sampling_rate = 44100  # Типова частота дискретизації для аудіофайлів (Hz)
threshold = 10  # Поріг для виявлення сигналу
# === Завантаження реального аудіофайла ===
def load_audio(file_path):
    rate, data = wavfile.read(file_path)
    return rate, data
# === Аналіз сигналу ===
def analyze_signal(signal, rate):
    fft_spectrum = np.fft.fft(signal)
    freq = np.fft.fftfreq(len(fft_spectrum), d=1 / rate)
    # Повертаємо лише позитивні частоти
    positive_freqs = freq[freq > 0]
    positive_spectrum = np.abs(fft_spectrum[freq > 0])
    return positive_freqs, positive_spectrum
# === Виявлення частот цілей ===
def detect_targets(positive_freqs, positive_spectrum, threshold=5):
    # Знаходимо піки у спектрі
    peaks, _ = find_peaks(positive_spectrum, height=threshold)
    detected_frequencies = positive_freqs[peaks]
    return detected_frequencies
# === Основна програма ===
def main():
    file_path = "drone_sound.wav"  # Шлях до мого аудіофайла
    print(f"Завантаження аудіофайлу: {file_path}")
    # Завантаження аудіофайла
    rate, data = load_audio(file_path)
    # Якщо файл стерео, перетворюємо в моно
    if len(data.shape) > 1:
        data = data.mean(axis=1)
    print("Аналіз сигналу...")
    positive_freqs, positive_spectrum = analyze_signal(data, rate)
    print("Виявлення цілей...")
    detected_frequencies = detect_targets(positive_freqs, positive_spectrum, threshold)
    # Виведення результатів
    print("Виявлені частоти цілей (Гц):", detected_frequencies)
    for freq in detected_frequencies:
        if freq in target_frequencies:
            print(f"Ціль виявлена на частоті {freq} Гц!")
        else:
            print(f"Виявлено невідомий сигнал на частоті {freq} Гц.")
    # Побудова графіків
    plt.figure(figsize=(10, 6))
    # Сигнал у часі
    plt.subplot(2, 1, 1)
    time = np.arange(0, len(data)) / rate
    plt.plot(time, data, label="Сигнал")
    plt.title("Сигнал у часі")
    plt.xlabel("Час (с)")
    plt.ylabel("Амплітуда")
    plt.legend()
    # Частотний спектр
```

```
    plt.subplot(2, 1, 2)
    plt.plot(positive_freqs, positive_spectrum, label="Спектр сигналу")
    plt.scatter(
        detected_frequencies,
        positive_spectrum[[np.where(positive_freqs == f)[0][0] for f in detected_frequencies]],
        color="red", label="Виявлені цілі"
)
    plt.title("Частотний спектр сигналу")
    plt.xlabel("Частота (Гц)")
    plt.ylabel("Амплітуда")
    plt.legend()
    plt.tight_layout()
    plt.show()
# === Запуск програми ===
if __name__ == "__main__":
```

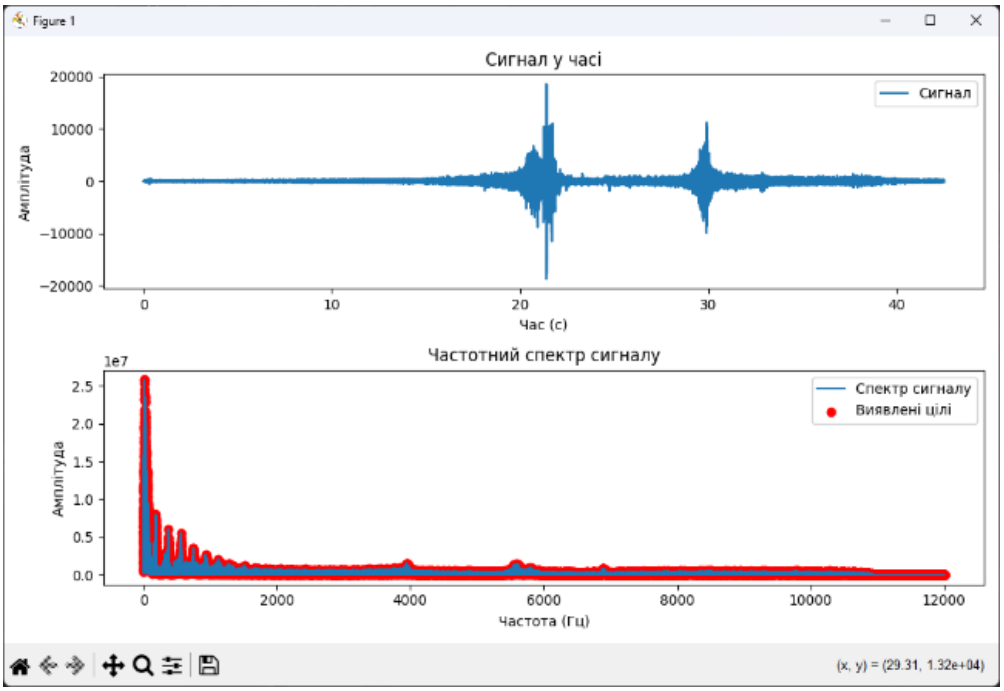

Рисунок 1 – Графіки представлення результатів програми

Алгоритм успішно ідентифікує частоти, відповідні цільовим характеристикам. Візуалізація у вигляді графіків демонструє точність аналізу: сигнал у часовій області і спектральний склад сигналу дають змогу оцінити як параметри сигналу, так і ефективність виявлення.

Висновки. Розроблений метод обробки сигналів довів свою практичну цінність у завданнях виявлення повітряних цілей. Його гнучкість забезпечується модульністю реалізації та можливістю розширення функціоналу. Подальше вдосконалення може включати адаптацію алгоритму до шумових середовищ, підвищення чутливості виявлення шляхом оптимізації порогів, а також інтеграцію методів класифікації цілей.

Список використаних джерел

1. Методика прогнозу тенденцій розвитку засобів військової радіолокації виявлення повітряних цілей. DOI: 10.30748/nitps.2021.43.15 (дата звернення 27.11.2024).
2. Verbytskyi I. V. Швидке перетворення Фур'є модульованих сигналів, представлених рядом Фур'є двох змінних. *Вісник Національного технічного університету «ХПІ». Серія: Нові рішення у сучасних технологіях*. 2018 № 16(1292). С. 102–106. DOI: 10.20998/2413-4295.2018.16.15 (дата звернення 26.11.2024).
3. The Creation of a Hidden Radar Field for the Detection of Small Aerial Objects due to the Use of Signals from Telecommunication Systems / Khudov et. al. *International Journal of Applied Engineering and Technology*. 2023. № 5(3). P. 9–17.

УДК 517.9:004.43(043.2)

Хріпко О. І., здобувач вищої освіти,
Фриз І. В., канд. фіз.-мат. наук, старший викладач кафедри інформаційних технологій

РОЗВ'ЯЗУВАННЯ ДИФЕРЕНЦІАЛЬНИХ РІВНЯНЬ НА МОВІ PYTHON

Донецький національний університет імені Василя Стуса, м. Вінниця

Сьогодні існують різноманітні технології, за допомогою яких обробка значних об'ємів даних проходить практично миттєво. Важливу роль відіграє математичний апарат, який є одним із базових інстументів для дослідження і моделювання складних систем. Зокрема, потужним математичним інструментарієм є диференціальні рівняння, які використовуються для математичного моделювання динамічних систем, під час дослідження і прогнозування реальних фізичних, біологічних, екологічних, соціально-економічних та інших процесів і явищ.

Ефективне розв'язування та дослідження диференціальних рівнянь і їх систем стало можливим завдяки різним інтерфейсам та бібліотекам, які забезпечують засобами для розв'язування диференціальних рівнянь, зокрема і за допомогою потужних модулів на базі Python. Отже, метою є вивчення можливостей для розв'язування та графічної візуалізації диференціальних рівнянь, зокрема за допомогою функції odeint бібліотеки SciPy і бібліотеки MatPlotLib.

Бібліотека SciPy, зокрема і її функція *odeint,* – це бібліотека, що надає користувачу базу для чисельного розв'язування диференціальних рівнянь [1], а бібліотека MatPlotLib містить інструментарій для візуалізації отриманих розв'язків. Інтерфейс odeint є ефективним для розв'язування звичайних диференціальних рівнянь, однак не завжди може бути застосованим для інших типів рівнянь.

На початку роботи під час розв'язування диференціальних рівнянь потрібно встановити необхідні бібліотеки: SciPy, NumPy, MatPlotLib. Розглянемо використання цих бібліотек [2] на прикладі диференціального рівняння із заданими початковими умовами для декількох значень параметра k:

$$\frac{dy(t)}{dt} = -k * y(t) \qquad (1)$$