



Учебна дисциплина: *Основи на инженерното проектиране*

ЛАБОРАТОРНО УПРАЖНЕНИЕ № 9

Тема: *Софтуерно дефинирани модели за генериране на аудио сигнали*

Продължителност: *3 учебни часа*

Задание:

- 1. Запознаване с методи за генериране на базови аудио сигнали.*
- 2. Изчисляване и визуализация на спектъра на сигналите.*
- 3. Създаване на музикална композиция.*

Цел: *Целта на лабораторното упражнение е студентите да се научат да създават, възпроизвеждат, визуализират и анализират аудио сигнали с различни форми (синусоидални, правоъгълни, асиметрични и симетрични триъгълни), да изчислят и визуализират спектъра им, както и да създават музикална композиция, използвайки Python.*



1. Теоретична част

1.1. Синусоидални сигнали

- Представяне: Синусоидалният сигнал е основен вид аудио сигнал, който има форма, подобна на синусоида. Той се характеризира с гладка и непрекъсната промяна на амплитудата и фазата във времето. Математически се описва чрез синусоидална функция.
- Приложение: Синусоидалните сигнали се използват в аудио сигналната обработка за различни цели, включително синтез на звук, модулация, детекция на честотата и фазата, както и за създаване на акустични ефекти. Те също така се използват при анализ на аудио сигнали и за измерване на параметри като честота и амплитуда.

1.2. Правоъгълни сигнали

- Представяне: Правоъгълният сигнал се характеризира с бързо и рязко преминаване между две стойности на амплитудата във времето.
- Приложение: Правоъгълните сигнали се използват често в цифровата обработка на сигналите и комуникациите. Те могат да бъдат използвани и за предаване на бинарна информация (1 и 0), като сигналите преминават през филтри и модулатори, които позволяват предаване на данни по каналите на свързаност. Освен това правоъгълните сигнали се използват в цифровите схеми и в електрониката за различни приложения.

1.3. Триъгълни сигнали

- Представяне: Триъгълният сигнал има форма, подобна на равнобедрен триъгълник или близка до правоъгълен триъгълник в зависимост от това дали е симетричен или несиметричен, като амплитудата се увеличава и намалява линейно във времето.
- Приложение: Триъгълните сигнали се използват рядко в обработката на аудио сигнали и музикалната индустрия, но те могат да бъдат полезни за тестове и измервания за електронни уреди и схеми. Също така те се използват в синтезаторите и електронните музикални инструменти за създаване на разнообразни звуци и тонове.

1.4. Спектър на аудио сигнал

Спектърът на аудио сигнал е графично представяне на разпределението на честотите и амплитудите на сигнала. Честотата представлява броя на колебанията на звуковия сигнал за една секунда, а амплитудата е височината на вълната. Спектърът може да бъде използван за различни цели, като:

- Идентифициране на различните компоненти на звука. Например, спектърът на човешкия глас може да се използва за идентифициране на гласните съгласни и гласните.



- Измерване на честотната характеристика на даден аудио устройство. Например, спектърът може да се използва за измерване на честотната характеристика на усилвател.
- Редактиране на аудио сигнали. Например, спектърът на аудио сигнал може да се използва за премахване на нежелани шумове или за подобряване на качеството на звука.

Ето някои примери за спектъра на аудио сигнали: Спектърът на синусоидален сигнал има една спектрална съставка. Спектърът на музикален инструмент може да бъде сложен и да има множество спектрални съставки. Спектърът на човешки глас е също сложен и има множество спектрални съставки. Той може да бъде представен по различни начини. Най-често срещаният начин е да се използва спектрална плътност на мощността, която представлява графично представяне на мощността на аудио сигнала като функция на честотата. Спектърът на аудио сигнала може да се използва и при по-сложен анализ чрез изкуствен интелект.

1.5. Честота на дискретизация (sampling rate)

Честотата на дискретизация е важен параметър при формирането на аудио сигнали и представлява броя на отчетите на звуковия сигнал, които се вземат всяка секунда. Това е мярка за времевата разделителна способност или "плътност" на измерването на аудио сигнала и определя колко информация се записва или предава за всяка секунда. По-висока честота на дискретизация означава по-голяма точност при представянето на аналоговия звуков сигнал като цифров. За да се запише или предаде аудио сигнал в цифров формат, аналоговият сигнал (който е непрекъснат) се дискретизира чрез вземане на множество отчети (нива) през определени интервали от време. Честотата на дискретизация определя колко често се вземат тези отчети и, следователно, колко добре може да бъде представен оригиналният аналогов сигнал. Ето някои примери за честоти на дискретизация и тяхното приложение:

- 20 kHz: Минимална честота на дискретизация, необходима за възпроизвеждане на човешкия слух.
- 44,1 kHz: Честота на дискретизация, използвана в компактдискете.
- 96 kHz: Честота на дискретизация, използвана в професионалните аудио приложения.



2. Задачи за изпълнение

Изисквания

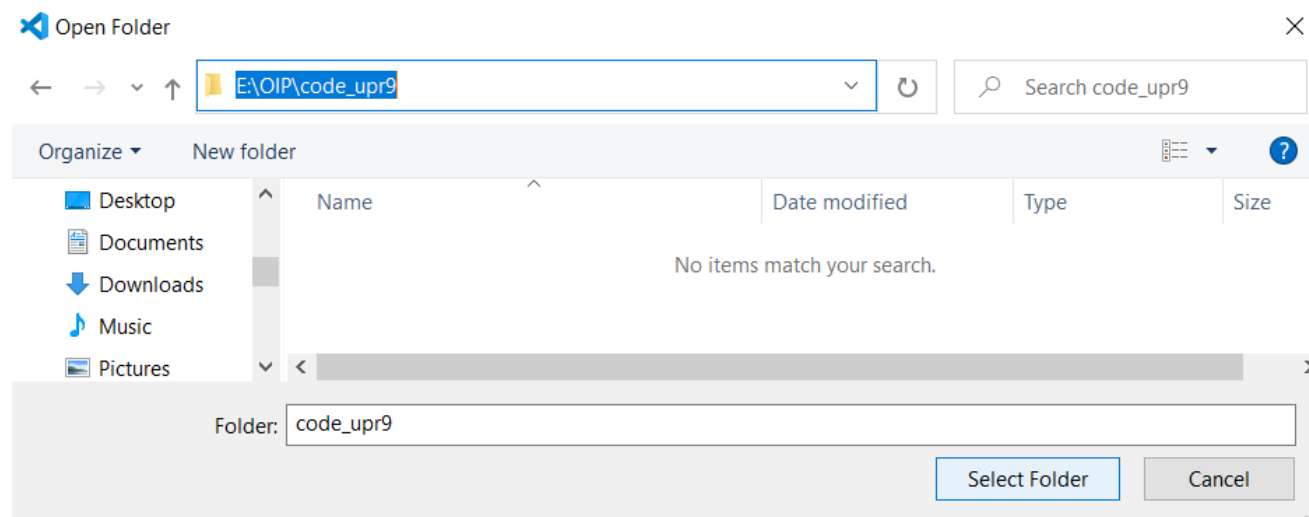
- Инсталиран Python 3.7 или по висока версия, примерна работна среда Visual Studio Code, библиотеките numpy, matplotlib, scipy и pyaudio.

Заб.: За всяка от описаните по-долу задачи използвайте файловете `audio_signal_synthesis_template.py` и `music_composition_template.py`, които представляват напътстващи образци на сорс кода, който трябва да напишете.

2.1. Генериране, възпроизвеждане и визуализация на базови аудио сигнали

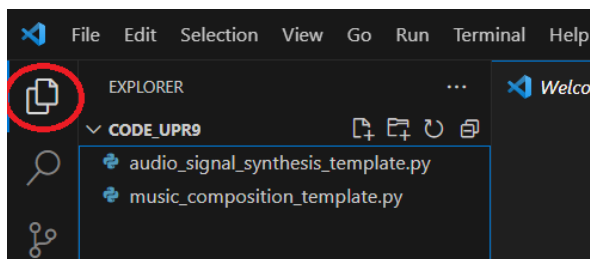
2.1.1. Подгответе средата Visual Studio Code за работа.

- Свалете придружаващите файлове `audio_signal_synthesis_template.py` и `music_composition_template.py` в папка на компютъра по Ваш избор (напр. `code_upr9`).
- Стартирайте Visual Studio Code и от менюто *File -> Open Folder...* отворете папката `code_upr9`, като навигирате до нея и изберете бутона *Select Folder*.



Фиг. 1. Отваряне на папка във Visual Studio Code

- Ако се появи съобщение *“Do you trust the authors of the files in this folder?”*, сложете отметка на *“Trust the authors of all files in the parent folder...”* и натиснете бутона *Yes, I trust the authors*.
- Ако файловете `audio_signal_synthesis_template.py` и `music_composition_template.py` не се появят автоматично в работното поле вляво, натиснете бутона ограден в червено от Фиг. 2.



Фиг. 2. Показване на работните файлове и папки в експлорера на Visual Studio Code

- Изберете файла *audio_signal_synthesis_template.py* като кликнете върху него еднократно с мишката, след което неговото съдържание трябва да се визуализира. Презапишете файла *audio_signal_synthesis_template.py* като *audio_signal_synthesis.py*. За целта, изберете *File -> Save As ... audio_signal_synthesis.py -> Save*.

2.1.2. В токущо създадения файл *audio_signal_synthesis.py* допишете програмата на Python така че тя да генерира аудио сигнал s във времето t : $s(t)$ със следните форми:

- Синусоидална (sinusoidal) (функцията *generate_sine_wave*):

$$s(t) = A \sin(2\pi f t). \quad (1)$$

- Правоъгълна (square) (функцията *generate_rectangular_wave*):

$$s(t) = A \operatorname{sign}(\sin(2\pi f t)). \quad (2)$$

- Асиметрична триъгълна (asymmetric triangle) (функцията *generate_asymetric_triangular_wave*):

$$s(t) = A \left(\frac{2}{T} (t \bmod T) - 1 \right). \quad (3)$$

- Симетрична триъгълна (symmetric triangle) (функцията *generate_symetric_triangular_wave*):

$$s(t) = 2A \left(1 - \frac{2}{T} \left| t \bmod T - \frac{T}{2} \right| \right) - 1, \quad (4)$$

където:

- A е амплитудата на сигнала;
- f е честотата на сигнала в Hz;
- T е периода на сигнала ($T=1/f$) в sec.;
- $\operatorname{sign}(x)$ е функция, която връща знака на аргумента x . Ако $x > 0$, $\operatorname{sign}(x)$ връща 1. Ако $x < 0$, $\operatorname{sign}(x)$ връща -1. Ако $x = 0$, функцията връща 0;
- \bmod е оператор делене по модул, който връща остатъка от делението на две числа. В python се използва %.



2.1.3. Допишете функцията за генериране на графика на сигналите във времето:

```
def visualize_signal(audio_signal, duration, title="Audio signal"):
```

За целта използвайте библиотеката `matplotlib`.

2.1.4. Допишете функцията за изчертаване на спектъра:

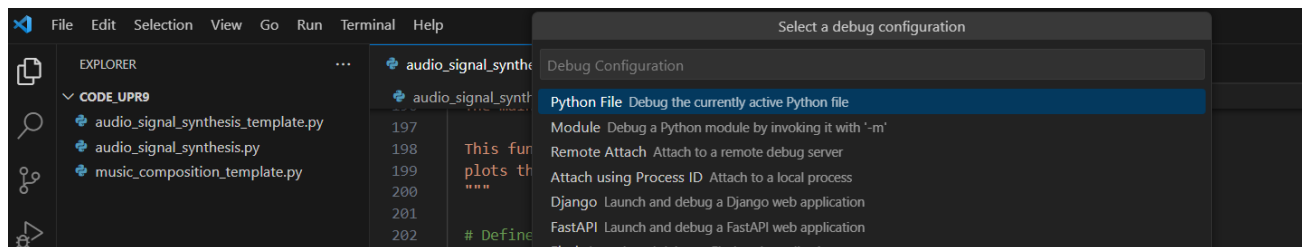
```
def plot_positive_spectrum(signal, title = "Signal Spectrum (Positive Frequencies Only)":
```

За целта използвайте библиотеката `matplotlib`.

2.1.5. От *main* функцията, за всяка форма на сигнала, задайте параметрите честота, продължителност и амплитуда. Честотата зависи от последната цифра на факултетния Ви номер (вж. файла *audio_signal_synthesis.py*).

2.1.6. За всеки от типовете базови сигнали, от *main* функцията, извикайте последователно функцията за генериране на съответния тип сигнал (напр. *generate_sine_wave()*), функцията за визуализация на сигнала (*visualize_signal()*), функцията за изчертаване на спектъра (*plot_positive_spectrum()*) и функцията за съхраняване на сигнала във файл (*save_signal_to_wav()*).

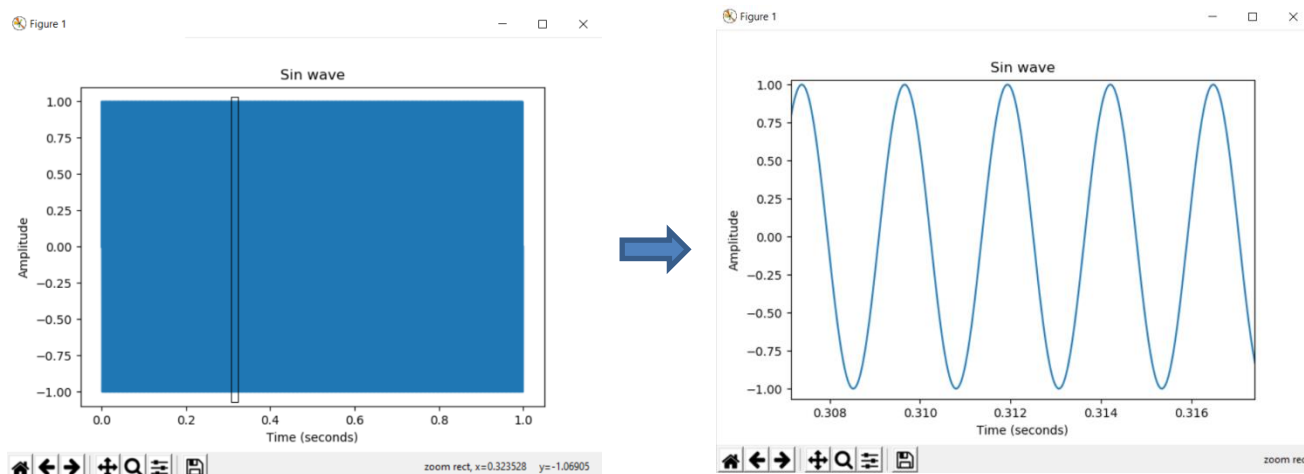
2.1.7. За да стартирате изпълнението на програмата, изберете *Run -> Start Debugging (F5)* (за режим на дебъгване) или *Run -> Run Without Debugging (Ctrl+F5)* (за режим без дебъгване). Препоръчва се по време на дописване на съответните функции да използвате режима с дебъгване. Ако се появи първоначално падащо меню за избор на конфигурацията за дебъгване, изберете *"Python File"* съгласно Фиг. 3.



Фиг. 3. Избор на конфигурация за дебъгване (Изберете *Python File*)

2.1.8. След като се визуализира съответната графика, при коректно изпълнение на програмата, използвайте инструмента "лупа", за да уголемите фрагмент от графиката, така че да се вижда добре формата на генерирания сигнал. Измерете периода и изчислете честотата на сигнала.

- Като отправен, можете да използвате примера на Фиг. 4 по-долу за случая на синусоидален сигнал



Фиг. 4. Визуализация на синусоиден сигнал

2.1.9. В протокола представете екранна снимка на всеки уголемен фрагмент за всеки от генерираните сигнали. Напишете отчетения период и изчисленията за честотата за всеки сигнал.

2.1.10. В протокола представете екранна снимка на спектъра за всеки от сигналите и напишете отчетената честота и амплитуда на основния сигнал и първите три хармоники (за синусоидалния сигнал няма допълнителни хармоници).

2.1.11. Всеки един от генерираните сигнали може да бъде възпроизведен с помощта на *VLC Media Player*. За целта отворете *VLC Media Player*, и от менюто *Media -> Open File*, навигирайте до директорията с работните файлове за упражнението (напр. *code_upr9* за примера от това ръководство) и отворете съответния файл (напр. *sin_wave.wav*).

2.2. Композиране на музика

2.2.1. Създайте програма на Python, която може да синтезира и възпроизвежда музикална композиция използвайки няколко ноти. За целта, отворете сорс кода от образаца *music_composition_template.py* и го презапишете като *music_composition.py*. Допишете програмата.

2.2.2. Музикалната композиция е последователност от ноти с определена честота, продължителност и амплитуда.

2.2.3. След успешно изпълнение, музикалната композиция може да бъде възпроизведена чрез *VLC Media Player*. За целта отворете *VLC Media Player*, и от менюто *Media -> Open File*, навигирайте до директорията с работните файлове за упражнението (напр. *code_upr9* за примера от това ръководство) и отворете файла синтезияния файл *composed_music.wav*.



3. Изводи

В протокола напишете изводи и заключения от лабораторното упражнение като обобщите какво сте научили и какви резултати сте получили от анализа на аудио сигналите и композирането на музика.