Sprawozdanie z laboratorium

Przetwarzanie Sygnałów i Obrazów

Ćwiczenie numer: 1

Temat: Wprowadzenie

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

08.10.2021r.

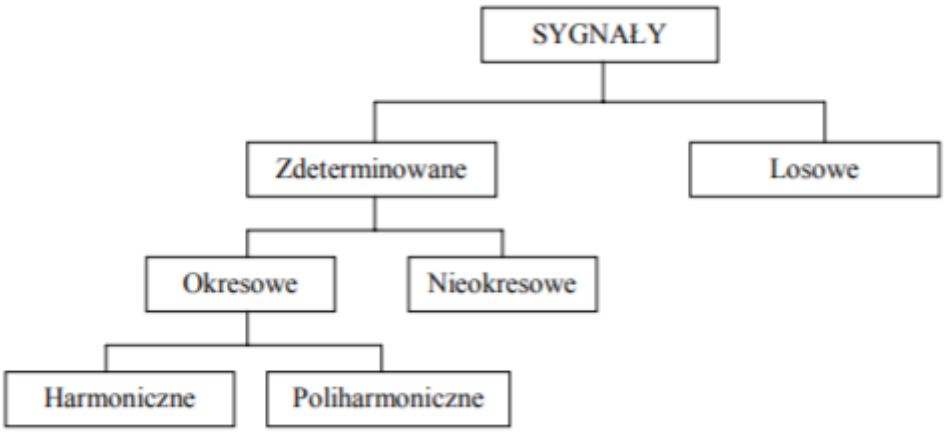
.....

Data i podpis prowadzącego

Wprowadzenie

Przed przystąpieniem do wykonania zadań wprowadzających wybrano i przygotowano środowisko programistyczne. Postawiono na Visual Studio Code. Niezbędne było pobranie rozszerzenia Pythona dla VSC oraz zainstalowanie bibliotek Numpy, Matplotlib i Scipy.

Tematem zajęć były proste sygnały. Jak zatem można zdefiniować to pojęcie? Sygnał jest nośnikiem informacji o tym jak dowolna, mierzalna wielkość zmienia się w czasie. Sygnały można podzielić na losowe, zdeterminowane, okresowe, nieokresowe, harmoniczne i poliharmoniczne. [1,2]



Rysunek 1 Podstawowa klasyfikacja sygnałów[2]

Zadanie 1

Treść zadania:

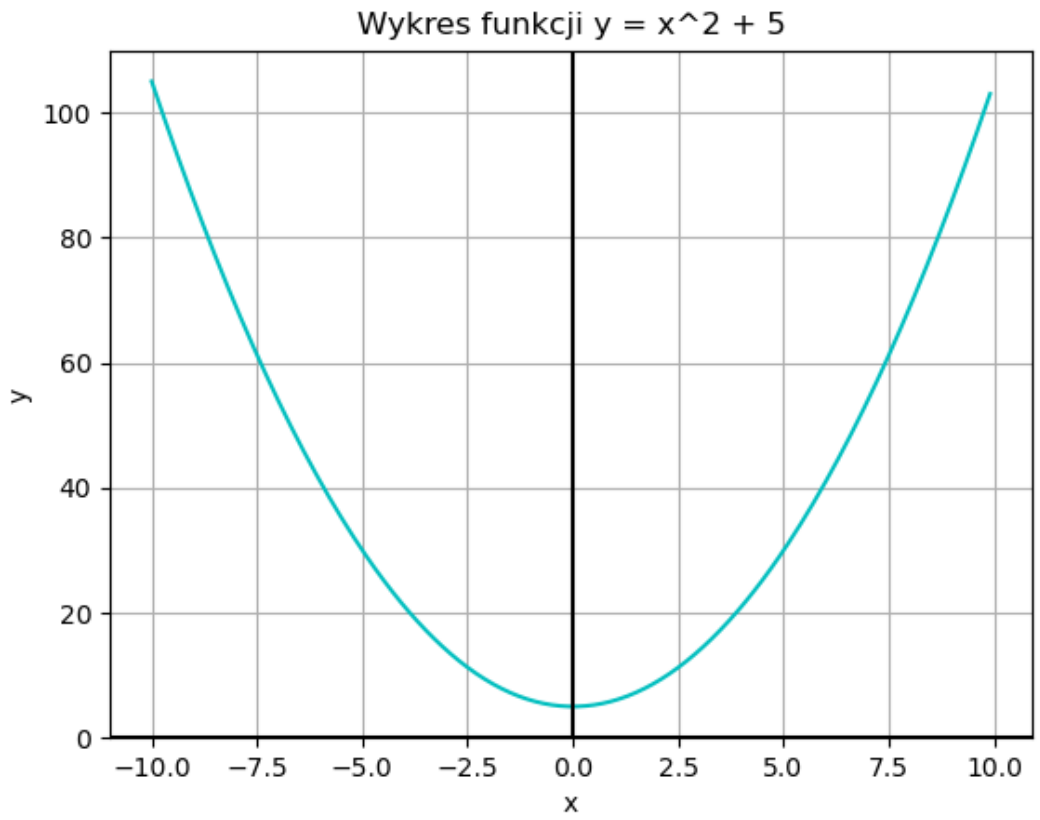
Sporządź wykres funkcji y = x^2 + 5, w dziedzinie od −10 do 10. Wykorzystać funkcję arange lub linspace z biblioteki Numpy.

Realizacja w kodzie

```
1   import numpy as np  # Biblioteka numpy jako np
2   import matplotlib.pyplot as plt # Biblioteka matplotlib od python plot jako plt
3
4   x=np.arange(-10,10,0.1) # Zakres x od -10 do 10
5   y=pow(x,2)+5 # wzór funkcji
6
7   plt.plot(x,y,'c') # Rysowanie w zakresie zmiennej x funkcji y, 'c' oznacza linie ciągłą
8   plt.axhline(y=0,color = "k") # Zaznacznie lini y=0 kolorem czarnym
9   plt.axvline(x=0,color = "k") # Zaznacznie lini x=0 kolorem czarnym
10  plt.grid(True,which='both') # Włączenie siatki
11  plt.title("Wykres funkcji y = x^2 + 5") # Tytuł wykresu
12  plt.xlabel("x") # Podpis osi x
13  plt.ylabel("y") # Podpis osi y
14  plt.show() # Wywołanie wykresu
```

Rysunek 2 Kod programu [Screen ekranu]

Wynik



Wykres 1 Wykres wygenerowany przez program [Screen ekranu]

Zadanie 2

Treść zadania:

Na jednym rysunku sporządź wykres funkcji sin(x) oraz cos(x), w dziedzinie od 0 do 2π.

Realizacja w kodzie

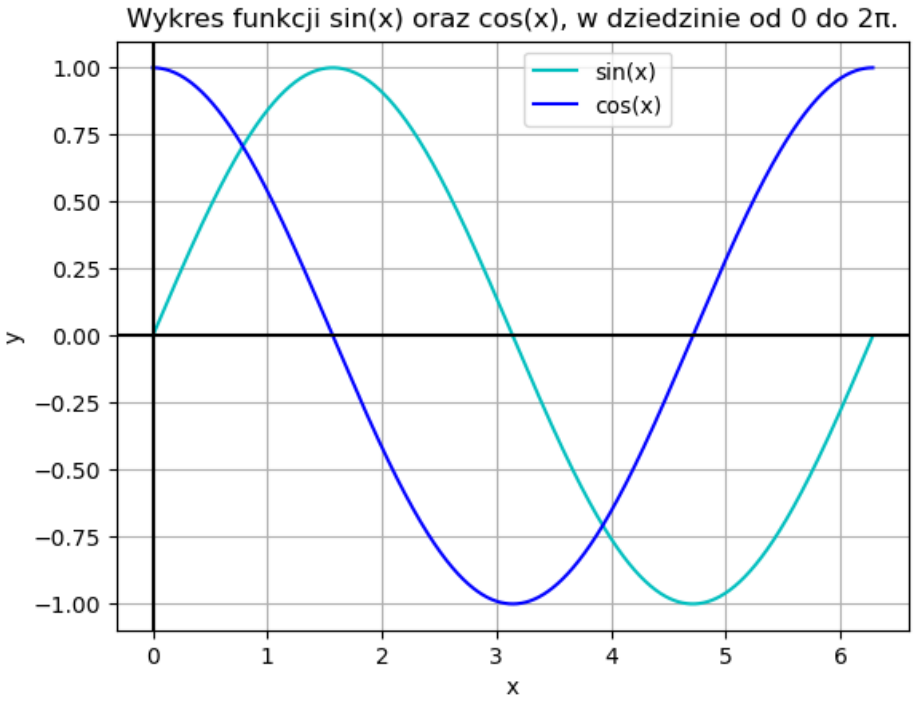
```
import numpy as np  # Biblioteka numpy jako np
import matplotlib.pyplot as plt # Biblioteka matplotlib od python plot jako plt

x = np.linspace(0, 2 * np.pi, 10000) # Określenie dziedziny <0,2π>, 10.000 pomiarów w tym zakresie
y1 = np.sin(x) # Wzór funkcji y1
y2 = np.cos(x) # Wzór funkcji y2

plt.plot(x,y1,'c') # Rysowanie w zakresie zmiennej x funkcji y1, 'c' oznacza linie ciągłą
plt.plot(x,y2,'b') # Rysowanie w zakresie zmiennej x funkcji y2, 'c' oznacza linie ciągłą
plt.legend(['sin(x)','cos(x)'],bbox_to_anchor=(0.5, 1)) # Stworzenie legendy i ustalenie jej lokalizacji na wykresie
plt.axhline(y=0,color = "k") # Zaznacznie lini y=0 kolorem czarnym
plt.axvline(x=0,color = "k") # Zaznacznie lini x=0 kolorem czarnym
plt.grid(True,which='both') # Włączenie siatki
plt.title("Wykres funkcji sin(x) oraz cos(x), w dziedzinie od 0 do 2π.") # Tytuł wykresu
plt.xlabel("x") # Podpis osi x
plt.ylabel("y") # Podpis osi y
plt.show() # Wywołanie wykresu
```

Rysunek 3 Kod programu [Screen ekranu]

Wynik



Wykres 2Wykres wygenerowany przez program [Screen ekranu]

Zadanie 3

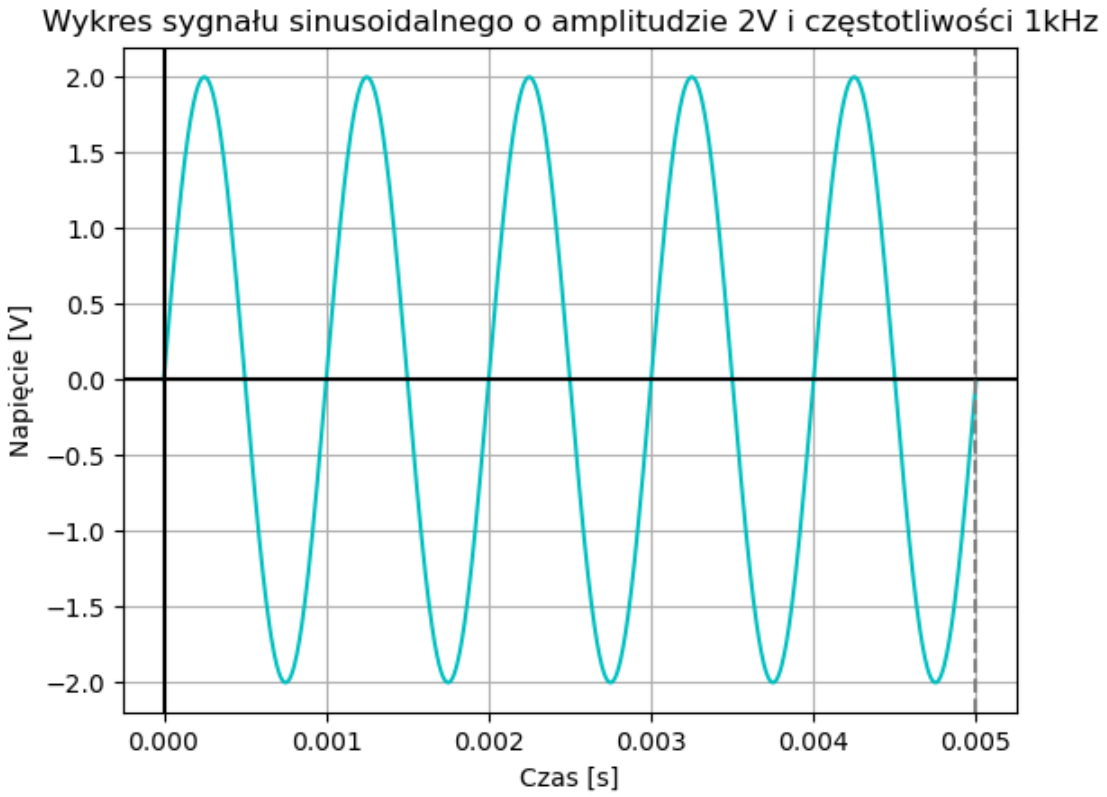
Treść zadania:
Sporządź wykres sygnału sinusoidalnego o częstotliwości 1kHz i amplitudzie 2V w zakresie od 0 do 5ms

Realizacja w kodzie

```
1   import numpy as np  # Biblioteka numpy jako np
2   import matplotlib.pyplot as plt # Biblioteka matplotlib od python plot jako plt
3
4   x = np.linspace(0, 0.005, 1000) # Określenie dziedziny <0,2π>, 1000 pomiarów w tym zakresie
5   y = 2 * np.sin(2*np.pi*1000*x) # Wzór funkcji
6
7   plt.plot(x,y,'c') # Rysowanie w zakresie zmiennej x funkcji y, 'c' oznacza linie ciągłą
8   plt.axhline(y=0,color = "k") # Zaznacznie lini y=0 kolorem czarnym
9   plt.axvline(x=0,color = "k") # Zaznacznie lini x=0 kolorem czarnym
10  plt.axvline(x=0.005,color = "grey",linestyle = "--") # Zaznaczeni lini x=0.005 kolorem szarym linią przerywaną
11  plt.grid(True,which='both') # Włączenie siatki
12  plt.title("Wykres sygnału sinusoidalnego o amplitudzie 2V i częstotliwości 1kHz") # Tytuł wykresu
13  plt.xlabel("Czas [s]") # Podpis osi x
14  plt.ylabel("Napięcie [V]") # Podpis osi y
15  plt.show() # Wywołanie wykresu
```

Rysunek 4 Kod programu [Screen ekranu]

Wynik



Wykres 3 Wykres wygenerowany przez program [Screen ekranu]

Zadanie 4

Treść zadania:

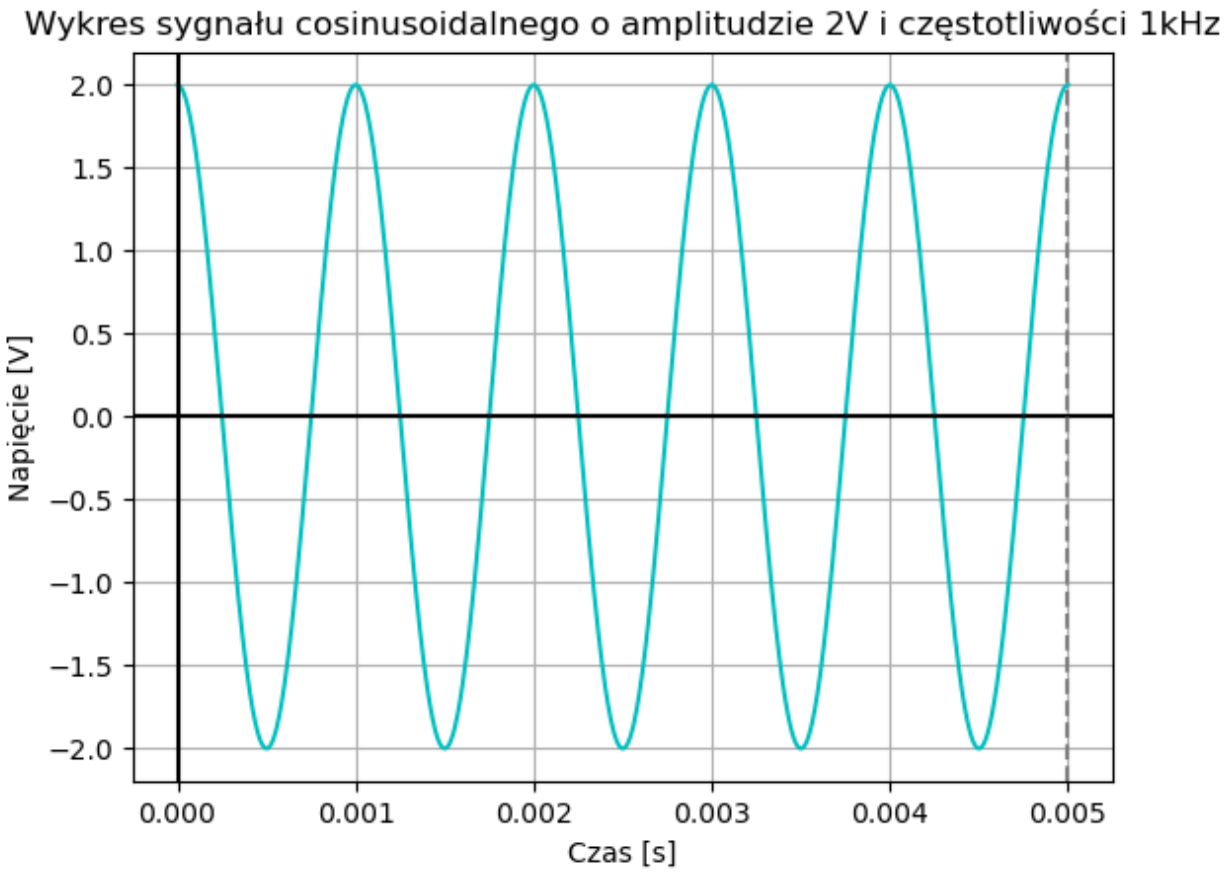
Powtórz Zadanie 1.3 dla sygnałów: a) kosinusoidalnego, b) prostokątnego, c) trójkątnego, d) piłokształtnego.

A) Realizacja w kodzie

```
1   import numpy as np  # Biblioteka numpy jako np
2   import matplotlib.pyplot as plt # Biblioteka matplotlib od python plot jako plt
3
4   x = np.linspace(0, 0.005, 1000) # Określenie dziedziny <0,2π>, 1000 pomiarów w tym zakresie
5   y = 2 * np.cos(2*np.pi*1000*x) # Wzór funkcji
6
7   plt.plot(x,y,'c') # Rysowanie w zakresie zmiennej x funkcji y, 'c' oznacza linie ciągłą
8   plt.axhline(y=0,color = "k") # Zaznacznie lini y=0 kolorem czarnym
9   plt.axvline(x=0,color = "k") # Zaznacznie lini x=0 kolorem czarnym
10  plt.axvline(x=0.005,color = "grey",linestyle = "--") # Zaznaczeni lini x=0.005 kolorem szarym linią przerywaną
11  plt.grid(True,which='both') # Włączenie siatki
12  plt.title("Wykres sygnału cosinusoidalnego o amplitudzie 2V i częstotliwości 1kHz") # Tytuł wykresu
13  plt.xlabel("Czas [s]") # Podpis osi x
14  plt.ylabel("Napięcie [V]") # Podpis osi y
15  plt.show() # Wywołanie wykresu
```

Rysunek 5 Kod programu [Screen ekranu]

Wynik



Wykres 4 Wykres wygenerowany przez program [Screen ekranu]

B) Realizacja w kodzie:

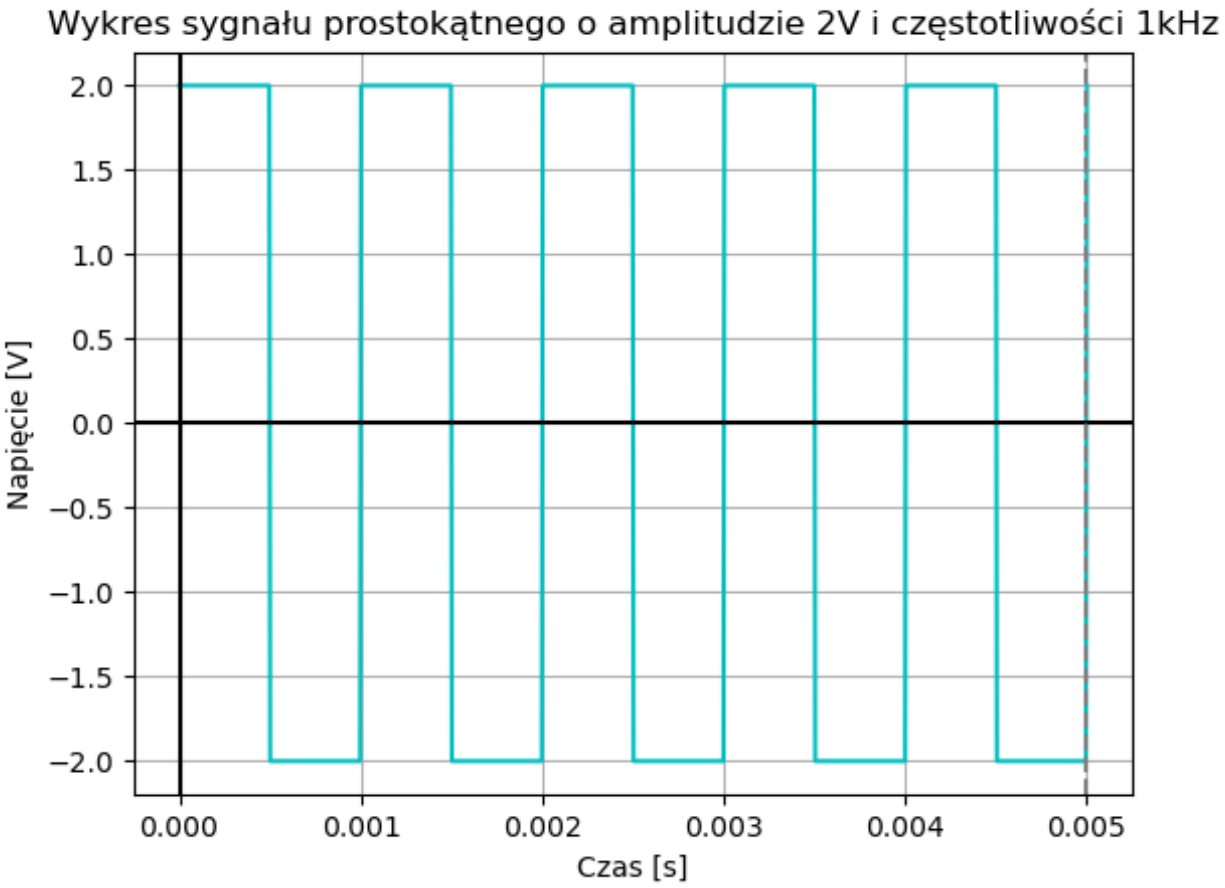
```
import numpy as np  # Biblioteka numpy jako np
import matplotlib.pyplot as plt # Biblioteka matplotlib od python plot jako plt
from scipy import signal as si # Biblioteka scipy importowanie signala jako si

x = np.linspace(0, 0.005, 1000) # Określenie dziedziny <0,2π>, 1000 pomiarów w tym zakresie
y = 2 * si.square(2*np.pi*1000*x) # Wzór funkcji

plt.plot(x,y,'c') # Rysowanie w zakresie zmiennej x funkcji y, 'c' oznacza linie ciągłą
plt.axhline(y=0,color = "k") # Zaznacznie lini y=0 kolorem czarnym
plt.axvline(x=0,color = "k") # Zaznacznie lini x=0 kolorem czarnym
plt.axvline(x=0.005,color = "grey",linestyle = "--") # Zaznaczeni lini x=0.005 kolorem szarym linią przerywaną
plt.grid(True,which='both') # Włączenie siatki
plt.title("Wykres sygnału prostokątnego o amplitudzie 2V i częstotliwości 1kHz") # Tytuł wykresu
plt.xlabel("Czas [s]") # Podpis osi x
plt.ylabel("Napięcie [V]") # Podpis osi y
plt.show() # Wywołanie wykresu
```

Rysunek 6 Kod programu [Screen ekranu]

Wynik



Wykres 5 Wykres wygenerowany przez program [Screen ekranu]

C) Realizacja w kodzie

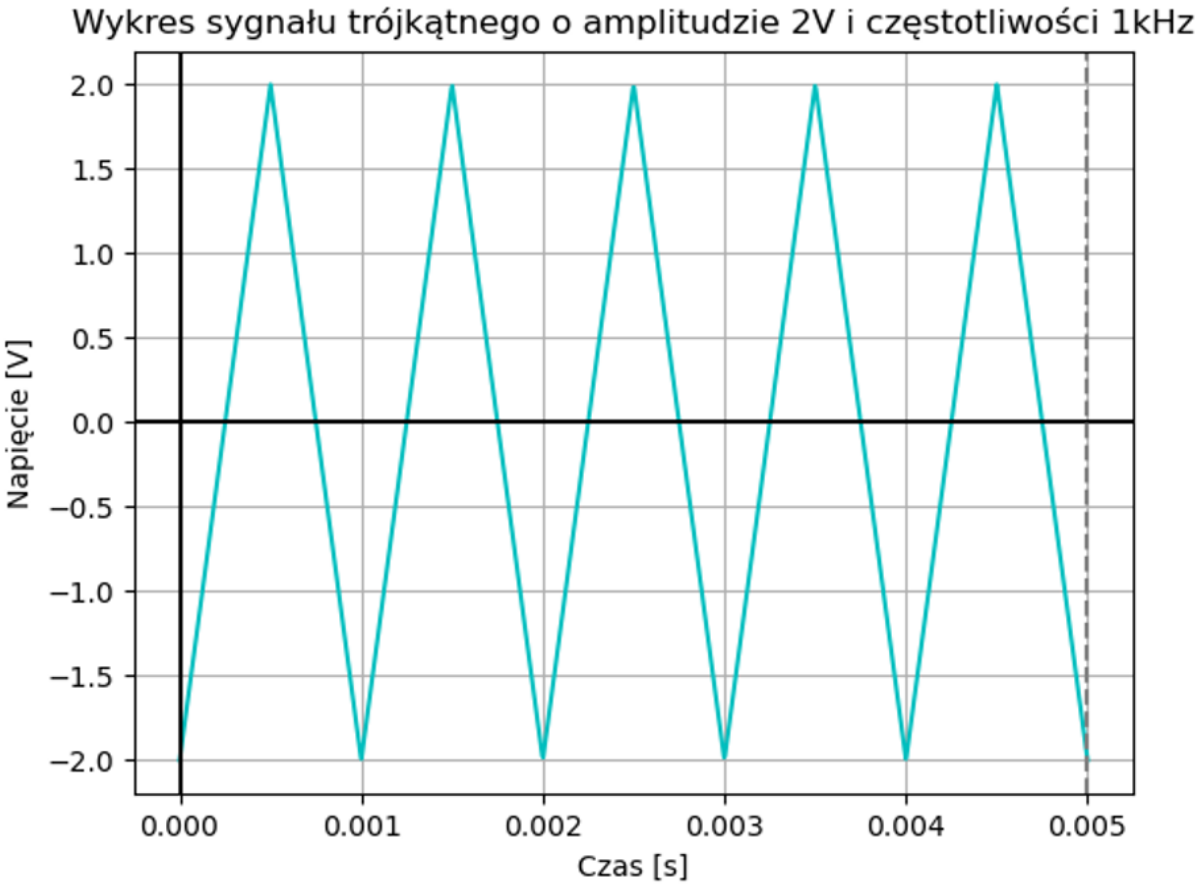
```
import numpy as np  # Biblioteka numpy jako np
import matplotlib.pyplot as plt # Biblioteka matplotlib od python plot jako plt
from scipy import signal as si # Biblioteka scipy importowanie signala jako si

x = np.linspace(0, 0.005, 1000) # Określenie dziedziny <0,2π>, 1000 pomiarów w tym zakresie
y = 2 * si.sawtooth(2*np.pi*1000*x,0.5) # Wzór funkcji z określeniem sygnały piłokształtnego na trójkątny

plt.plot(x,y,'c') # Rysowanie w zakresie zmiennej x funkcji y, 'c' oznacza linie ciągłą
plt.axhline(y=0,color = "k") # Zaznacznie lini y=0 kolorem czarnym
plt.axvline(x=0,color = "k") # Zaznacznie lini x=0 kolorem czarnym
plt.axvline(x=0.005,color = "grey",linestyle = "--") # Zaznaczeni lini x=0.005 kolorem szarym linią przerywaną
plt.grid(True,which='both') # Włączenie siatki
plt.title("Wykres sygnału trójkątnego o amplitudzie 2V i częstotliwości 1kHz") # Tytuł wykresu
plt.xlabel("Czas [s]") # Podpis osi x
plt.ylabel("Napięcie [V]") # Podpis osi y
plt.show() # Wywołanie wykresu
```

Rysunek 7 Kod programu [Screen ekranu]

Wynik



Wykres 6 Wykres wygenerowany przez program [Screen ekranu]

D) Realizacja w kodzie

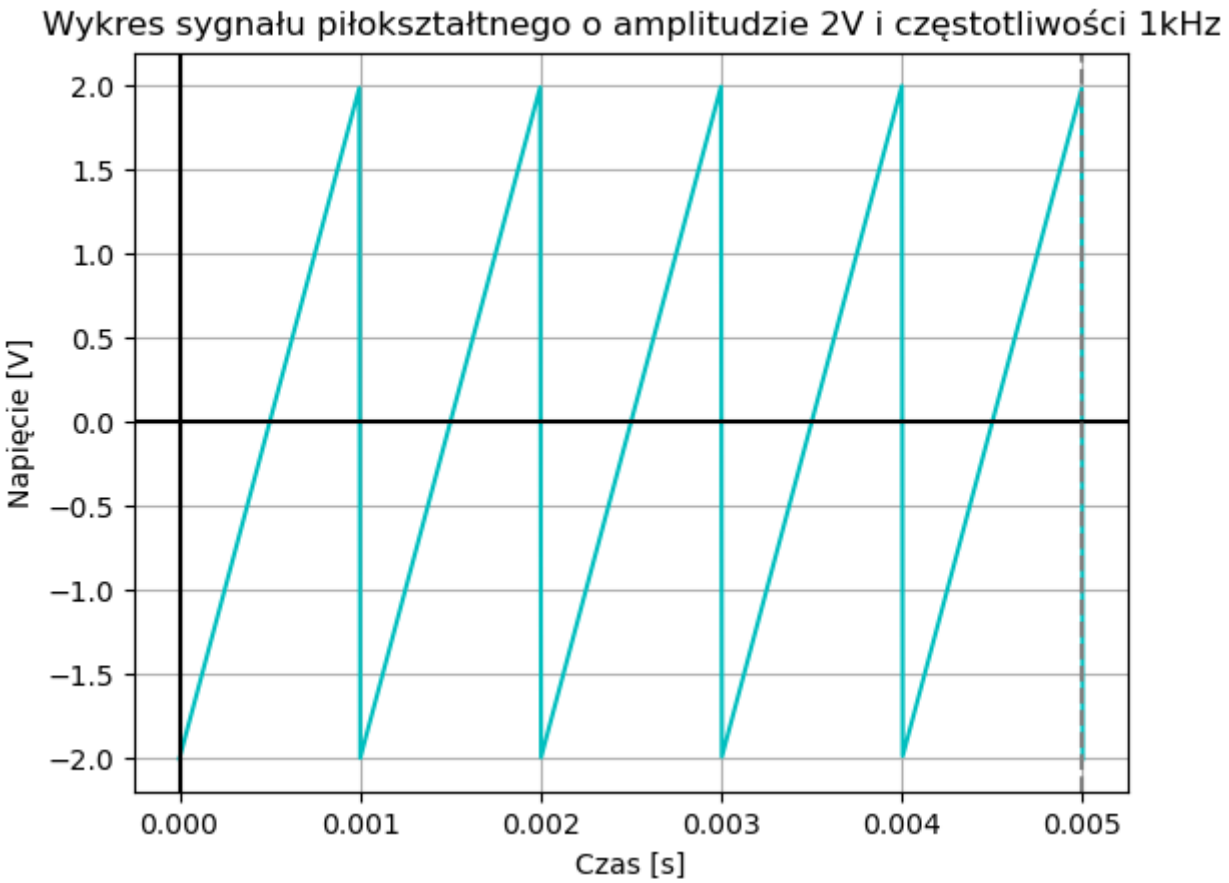
```python
import numpy as np  # Biblioteka numpy jako np
import matplotlib.pyplot as plt # Biblioteka matplotlib od python plot jako plt
from scipy import signal as si # Biblioteka scipy importowanie signala jako si

x = np.linspace(0, 0.005, 1000) # Określenie dziedziny <0,2π>, 1000 pomiarów w tym zakresie
y = 2 * si.sawtooth(2*np.pi*1000*x) # Wzór funkcji

plt.plot(x,y,'c') # Rysowanie w zakresie zmiennej x funkcji y, 'c' oznacza linie ciągłą
plt.axhline(y=0,color = "k") # Zaznacznie lini y=0 kolorem czarnym
plt.axvline(x=0,color = "k") # Zaznacznie lini x=0 kolorem czarnym
plt.axvline(x=0.005,color = "grey",linestyle = "--") # Zaznaczeni lini x=0.005 kolorem szarym linią przerywaną
plt.grid(True,which='both') # Włączenie siatki
plt.title("Wykres sygnału piłokształtnego o amplitudzie 2V i częstotliwości 1kHz") # Tytuł wykresu
plt.xlabel("Czas [s]") # Podpis osi x
plt.ylabel("Napięcie [V]") # Podpis osi y
plt.show() # Wywołanie wykresu
```

Rysunek 8 Kod programu [Screen ekranu]

Wynik



Wykres 7 Wykres wygenerowany przez program [Screen ekranu]

Zadanie 5

Treść zadania

Samodzielnie przestudiować metody tworzenia i modyfikowania macierzy z użyciem biblioteki Numpy.

Realizacja zadania

```
1   import numpy as nu;
2
3   A = nu.array([[1,2,3],[1,2,3]])
4   B = nu.array([[1,1,1],[1,1,1]])
5   suma = nu.array([0]*3 for i in range (2)) # 3 kolumny 2 wiersze macierz zerowa
6   print("Dodawanie macierzy A i B")
7   suma = A+B
8   print(suma)
9   print("Odejmowanie macierzy A i B")
10  suma = A-B
11  print(suma)
```

```
PROBLEMS 3   OUTPUT   TERMINAL   DEBUG CONSOLE

PS C:\Users\magda\Desktop\sem3\PSIO\pytoon> python -u "c:\Users\magda\Desktop\sem3\PSIO\pytoon\macierz.py"
Dodawanie macierzy A i B
[[2 3 4]
 [2 3 4]]
Odejmowanie macierzy A i B
[[0 1 2]
 [0 1 2]]
```

Rysunek 9 Dodawanie i odejmowanie macierzy [Screen ekranu]

```python
import numpy as nu;

A = nu.array([[1,2,3],
              [3,4,5]])
C = nu.array([[2,2,2],
              [2,2,2],
              [2,2,2]])
iloczyn = A.dot(C)      #mnożenie macierzy
print(iloczyn)
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
PS C:\Users\magda> python -u "c:\Users\magda\Desktop\
[[12 12 12]
 [24 24 24]]
PS C:\Users\magda>
```

Rysunek 10 Mnożenie macierzy [Screen ekranu]

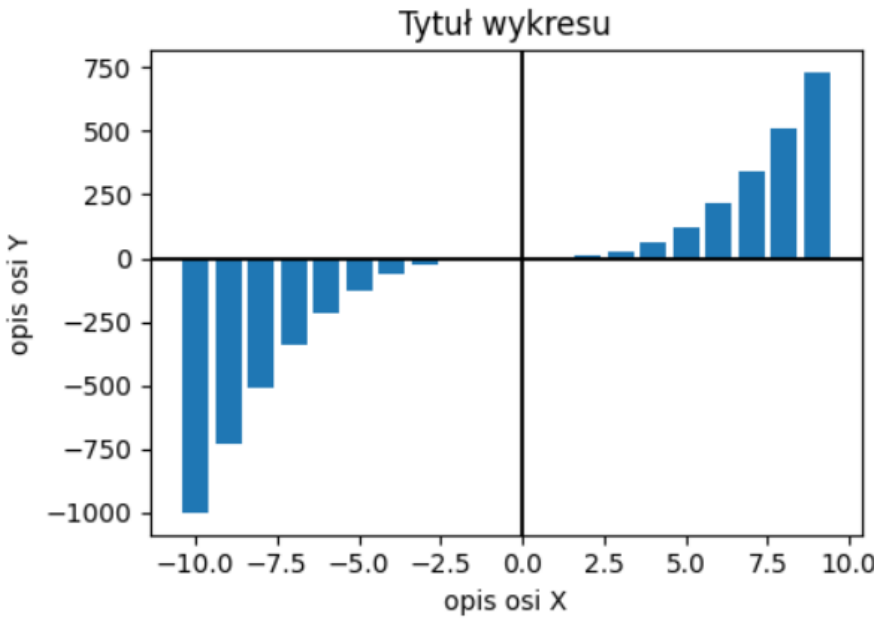
Zadanie 6

Treść zadania
Przestudiować materiały pomocnicze – Metody wizualizacji danych przy użyciu biblioteki Matplotlib.
Realizacja

```python
import numpy as np;
import matplotlib.pyplot as plt

x=np.arange(-10,10,1)
y=pow(x,3)

plt.bar(x,y)
plt.axhline(y=0,color="k")
plt.axvline(x=0,color = "k")
plt.title("Tytuł wykresu")
plt.xlabel("opis osi X")
plt.ylabel("opis osi Y")
plt.show()
```



Rysunek 11 Podstawowe funkcje do tworzenia wykresów, wykres słupkowy [Screen ekranu]

Zadanie 7

Treść zadania
Przestudiować materiały pomocnicze – Dokumentacja oprogramowania JupyterLab.
Realizacja(we wnioskach)

Podsumowanie i wnioski

Dzięki wykonanym zadaniom obeznano się z językiem Python. Za jego pomocą w łatwy sposób można tworzyć wykresy. Język jest stosunkowo łatwy. Niezbędne jednak okazały się biblioteki NumPy, Matplotlib, Scipy. Zawierają one wiele przydatnych funkcji takich jak wyznaczanie zakresu zmiennych, podpisywanie osi i wykresu, tworzenia legend i co najważniejsze, rysowania i wyświetlania wykresów. Funkcja *linspace()* dzięki możliwości ustawienia ilości punktów pomiarowych, daję imponującą aproksymację.

Wykonanie pierwszego zadanie gwarantuje zapoznanie się z niezbędnymi funkcjami z bibliotek Numpy oraz Matplotlib do tworzenia wykresów. Jak się okazało w zadaniu drugim, nie potrzeba dodatkowych funkcji, aby otrzymać jednocześnie dwa lub więcej wykresów. Funkcja *plot()* z biblioteki Matplotlib ma możliwość ustawienia koloru więc bez problemu można odróżnić od siebie wykresy. Można dodatkowo dodać legendę za pomocą funkcji *legend()* z tej samej biblioteki. Zadania trzecie i czwarte są bardzo podobne. Do ich wykonania potrzebna była dodatkowo znajomość podstawowych wzorów fizycznych. Sygnały prostokątny, trójkątni i piłokształtny wymagały użycia biblioteki Scipy. Dzięki zadaniu piątemu zapoznano się z tworzeniem i operowanie macierzami. Potrzebna do tego jest jedynie biblioteka Numpy. Jak się okazało użycie operatora ” * „ lub funkcji *multiply()* nie jest poprawnym mnożeniem macierzy. Dokładny wynik otrzymamy używając funkcji *dot().* W zadaniu szóstym należało przestudiować bibliotekę Matplotlib. Zawiera ona funkcje nie tylko do tworzenia regularnych wykresów, ale także punktowych, słupkowych, konturowych, kołowych lub 3D. A to tylko część z możliwości.

Zadanie siódme zawierało zapoznanie się z dokumentacją JupyterLab. Jest to środowisko programistyczne do pracy z kodem i danymi. Umożliwia on m.in. korzystanie z terminali, edytorów tekstu i przeglądarek plików. Można w nim uruchamiać pliki typu „.py”. Można go pobrać m.in. komendą *pip install jupyterlab*, i uruchomić w przeglądarce komendą *jupyter lab.* Jupyter pozwala na tworzenie niezależnych fragmentów kodu, czyli komórek, jednocześnie pozwala odwoływać się do nich. Automatycznie zapisuje kod co dwie minuty, dzięki temu użytkownik nie musi się martwić o potencjalną utratę programu. Jupyter ma szeroką gamę skrótów klawiszowych, najważniejsze z nich to:

- wykonanie kodu – **Shift + Enter,**
- tworzenie nowej komórki – **Esc + b,**
- wejście do trybu komend – **Esc,**
- wyjście z trybu komend – **Enter.** [3,4]

```
C:\Users\magda>pip install jupyterlab
Collecting jupyterlab
  Using cached jupyterlab-3.2.0-py3-none-any.whl (8.6 MB)
Collecting packaging
  Using cached packaging-21.0-py3-none-any.whl (40 kB)
Collecting jupyter-server~=1.4
  Using cached jupyter_server-1.11.1-py3-none-any.whl (393 kB)
Collecting tornado>=6.1.0
  Using cached tornado-6.1.tar.gz (497 kB)
Collecting ipython
  Using cached ipython-7.28.0-py3-none-any.whl (788 kB)
Collecting jupyterlab-server~=2.3
  Using cached jupyterlab_server-2.8.2-py3-none-any.whl (58 kB)
```

Rysunek 12 Instalacja JupyterLab

Źródła lub bibliografia lub podobnie

[1] https://pl.wikipedia.org/wiki/Sygnał
[2] http://dydaktyka.polsl.pl
[3] https://jupyterlab.readthedocs.io/en/stable/index.html
[4] https://analityk.edu.pl/jupyter-notebook-edytor-tekstu-dla-python/