PROJEKT 2

# NATALIA GÓRECKA s19322

SPIS TREŚCI

[NATALIA GÓRECKA s19322 1](#_Toc61371394)

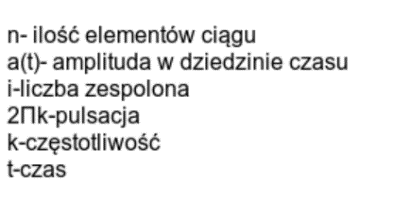
[1. Analiza częstotliwości-Transformata Fouriera 2](#_Toc61371395)

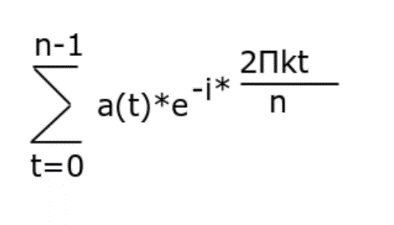
[2. Użyte biblioteki przy implementacji Transformaty Fouriera w Pythonie 2](#_Toc61371396)

[3. Implementacja Tranformaty Fouriera w Phytonie wraz z opisem 3](#_Toc61371397)

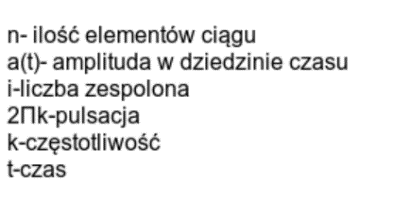
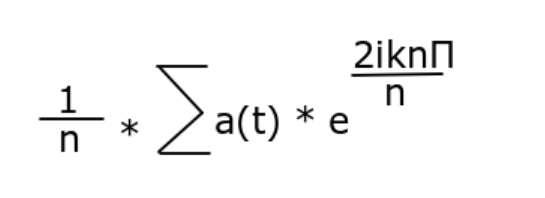
# Analiza częstotliwościowa-Transformata Fouriera

Transformata Fouriera przekształca skończony ciąg sygnałów rozdzielonych równomiernie od siebie w równy ciąg sygnałów. Odwrotna Transformata Fouriera jest szeregiem Fouriera wykorzystująca próbki Transformacji Fouriera jako współczynniki złożonych sinusoid przy odpowiednich częstotliwościach. Transformacja Fouriera pomaga w ,,wydobywaniu’’ pojedynczych sinusoid z sygnału wejściowego, który początkowo składa się z sinusoid sygnału jak i szumu.

Źródło: https://pl.wikipedia.org/wiki/Transformacja\_Fouriera



Wzór 1 Dyskretna Tranformata Fouriera



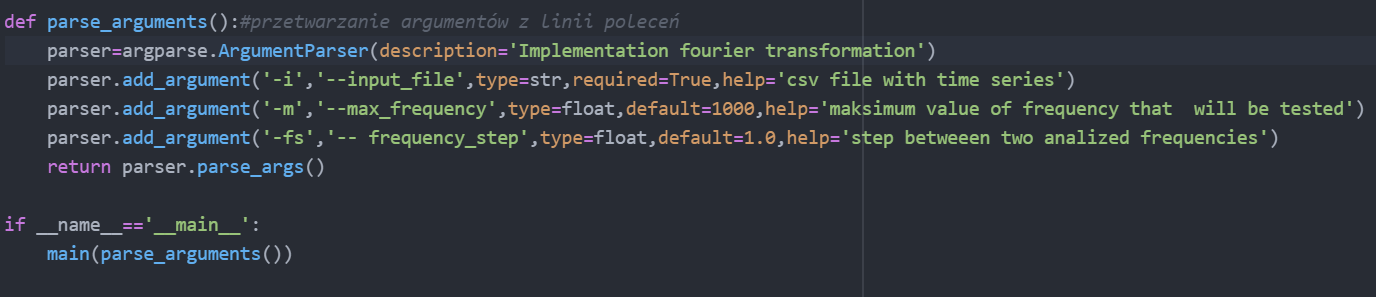
Wzór 2 Odwrotna Transformata Fouriera

# Użyte biblioteki wraz z metodami przy implementacji Transformaty Fouriera

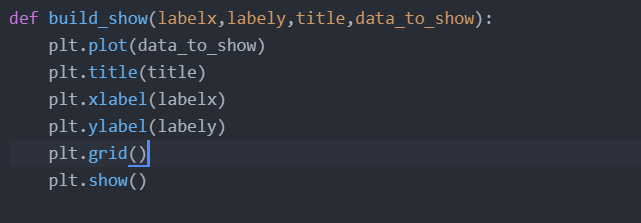
* Argparse
  + add\_argument()
  + parse\_args()
* Pandas
  + read.csv()
* Numpy
  + Asarray()
  + Dot()
  + Abs()
  + Arange()
  + Reshape()
  + Exp()
  + Copy()
  + Subtract()
* Matplotlib.pyplot
  + Plot()
  + Grid()
  + Xlabel()
  + Ylabel()
  + Show()

# Implementacja Tranformaty Fouriera w Phytonie

Do tego projektu został wgrany zestaw danych sygnałów wejściowych numer 2 przy pomocy biblioteki Pandas oraz metody read\_csv().



Do każdej wizualizacji używana jest zdefiniowana lokalnie metoda build\_show() używająca biblioteki Matplotlib.pyplot.



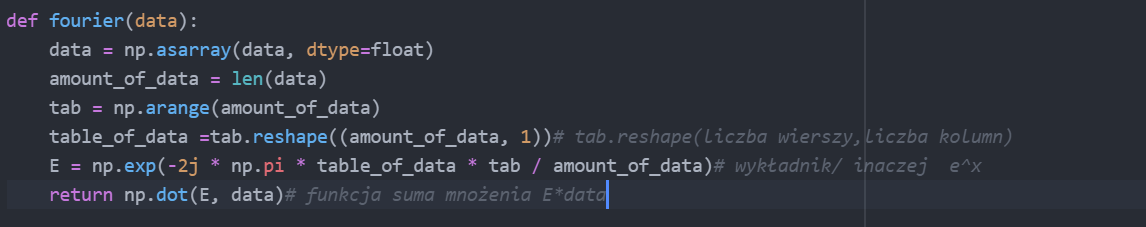
Pierwszym krokiem w projekcie po wgraniu sygnałów początkowych była wizualizacja ich przed jakimikolwiek zmianami na zestawie danych.

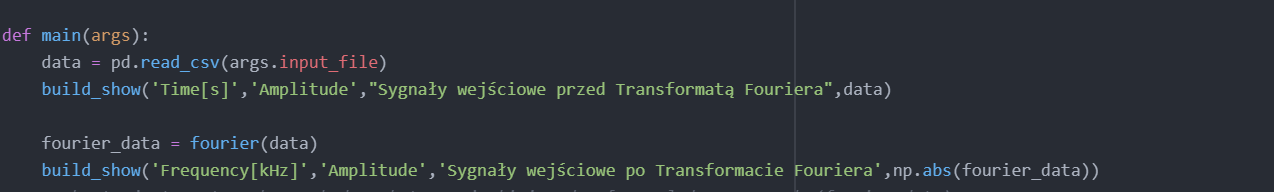
Chart

Description automatically generated

Rysunek 1 Sygnały wejściowe przed zastosowaniem Transformaty Fouriera

Sygnały zostały przekazane do metody fourier(), w której dane zostają zamienione na tablicę, następnie odczytana zostaje długość tablicy, która jest użyta w przedostatnim kroku. Uzyskana tablica jest wielowymiarowa o ilości wierszy równej wielkości danych oraz liczbie kolumn równej 1. W ostatnim kroku dane zostają poddane wzorowi tj.*Wzór 1*. Najpierw obliczony zostaje wykładnik e^x, a następnie zostaje on pomnożony przez każdy sygnał, a iloczyny zostają dodane do siebie dzięki jednej z metod biblioteki numpy dot(). W wyniku takich działań powstaje *Rysunek 2 Wizualizacja sygnałów wejściowych po zastosowaniu Transformaty Fouriera.*



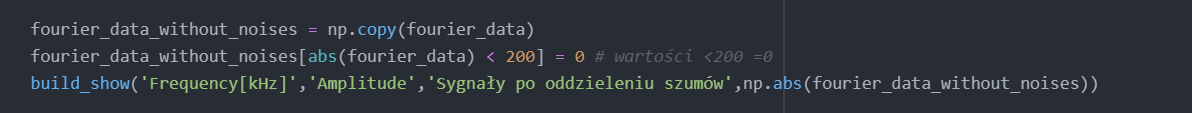


Chart

Description automatically generated

Rysunek 2 Wizualizacja sygnałów wejściowych po zastosowaniu Transformaty Fouriera

Wizualizacja jasno ukazuje sygnały wraz z szumami, które w kolejnym kroku zostają oddzielone na kopii tablicy fourier\_data. Wartości mniejsze od 200 zostają uznane jako szumy, a większe jako wartości sygnałów, dla których będą generowane sinusoidy *Rysunek 3 Wizualizacja Sygnałów po oddzieleniu szumów.*

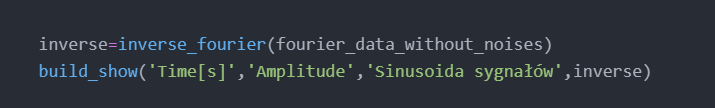


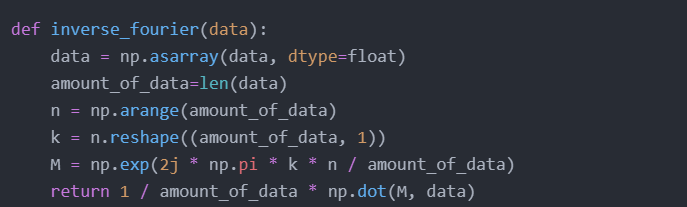
Chart

Description automatically generated

Rysunek 3 Wizualizacja sygnałów po oddzieleniu szumów

W metodzie main() wywołana jest funkcja inverse\_fourier(), do której przekazywane są dane bez szumów aby zastosować *odwrotną Transformatę Fouriera Wzór 2*, dzięki której uzyskamy sinusoidę sygnałów wejściowych bez szumów *Rysynek 4 Wizualizacja sinusoidy sygnałów bez szumów.*



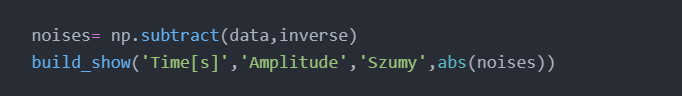


Chart, line chart

Description automatically generated

Rysunek 4 Wizualizacja sinusoidy sygnałów bez szumów

Ostatnim krokiem przy implementacji Transformaty jest wizualizacja samych szumów, które otrzymane zostały poprzez odjęcie pierwszej sinusoidy danych wejściowych od sinusoidy powstałej w wyniku wywołania metody inverse\_fourier().



Chart

Description automatically generated

Rysunek 5 Wizualizacja szumów