POLITECHNIKA KRAKOWSKA
WYDZIAŁ INFORMATYKI I TELEKOMUNIKACJI
KIERUNEK: INFORMATYKA

SZEREGI CZASOWE, GIEŁDA I EKONOMIA

DANE ARPA CHAIN VS WHITE NOISE

SPRAWOZDANIE Z LABORATORIUM

Maria Guz

Polecenia

Temat 1.

1. *Pobranie danych giełdowych.*
2. *Stworzenie wykresu pobranych danych w skali liniowej i pół-logarytmicznej.*
3. *Obliczenie logarytmicznych stóp zwrotu (logarithmic returns) pobranych danych oraz stworzenie ich wykresu.*
4. *Znormalizowanie szeregu stóp zwrotu (średnia równa 0 i odchylenie standardowe 1).*
5. *Wyrysowanie danych.*

Temat 2.

1. *Wygenerowanie danych o rozkładzie normalnym (white noise).*
2. *Stworzenie wykresu pobranych danych.*
3. *Policzenie sumy skumulowanej (ruch Browna).*
4. *Stworzenie wykresu dla sumy skumulowanej.*

Temat 3.

1. *Dla danych giełdowych (stopy zwrotu) i wygenerowanych policzenie histogramów.*
2. *Znormalizowanie histogramów (suma wartości słupków powinna być równa 1).*
3. *Policzenie parametrów rozkładów (skośności oraz kurtozy).*
4. *Przedstawienie obu znormalizowanych rozkładów na jednym wykresie.*

Temat 4.

1. *Dla danych giełdowych (stopy zwrotu) i wygenerowanych (biały szum) policzenie funkcji autokorelacji.*
2. *Policzenie funkcji autokorelacji dla modułu stóp zwrotu oraz modułu danych wygenerowanych. Sporządzenie odpowiedniego wykresu.*
3. *Policzenie widma mocy dla danych giełdowych (stopy zwrotu) i wygenerowanych (biały szum).*
4. *Policzenie widma mocy dla modułu stóp zwrotu oraz modułu danych wygenerowanych. Sporządzenie odpowiedniego wykresu.*

Wykonanie

Zadania zostały zrealizowane za pomocą języka programowania Python oraz dzięki środowisku PyCharm. Wybór tych narzędzi podyktowany był ich prejrzystością, intuicyjnością i prostotą. Do przeprowadzenia obliczeń wykorzystano studencki serwer obliczeń Torus za pośrednictwem JetBrains Gateway.

Dane

Analizowane dane dotyczą ceny ARPA Chain. Dane zostały pozyskane z pliku mat. i zapisane do pliku csv. Dane składają się z 912 393 rekordów, z których pierwsze 10 zostało zaprezentowane na grafice poniżej.

	0
0	0.01028
1	0.01026
2	0.01027
3	0.01026
4	0.01026
5	0.01026
6	0.01027
7	0.01027
8	0.01026
9	0.01026

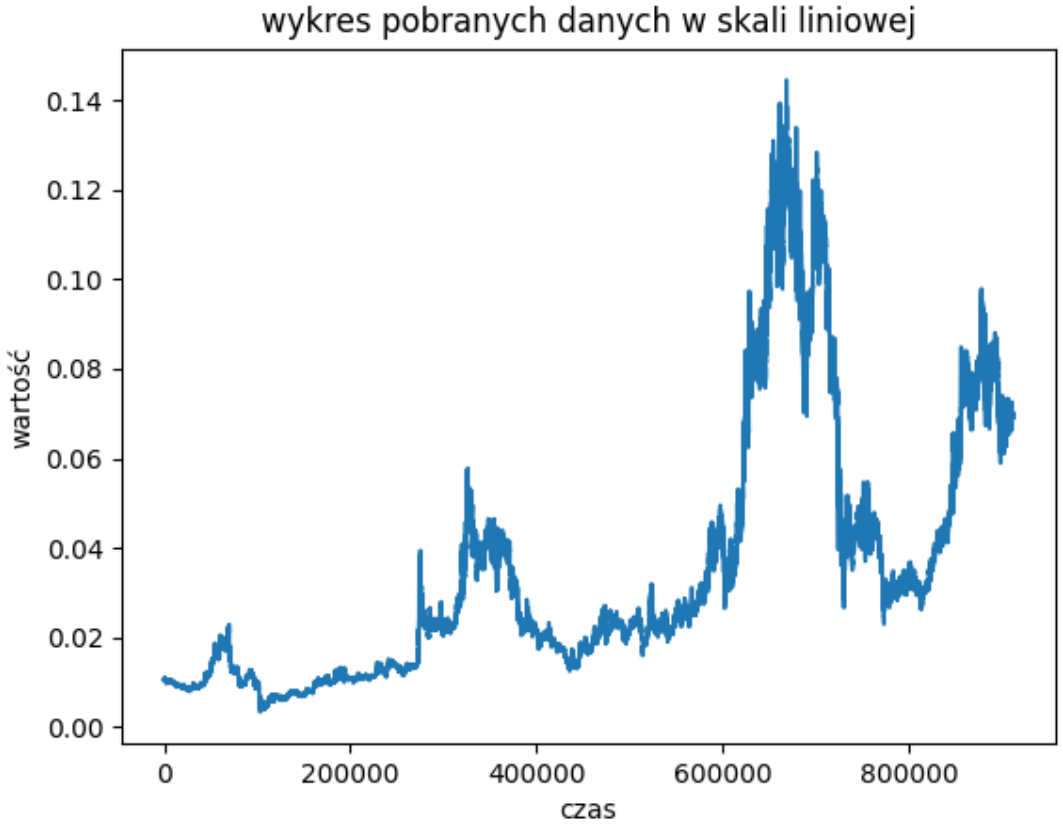
Temat 1.

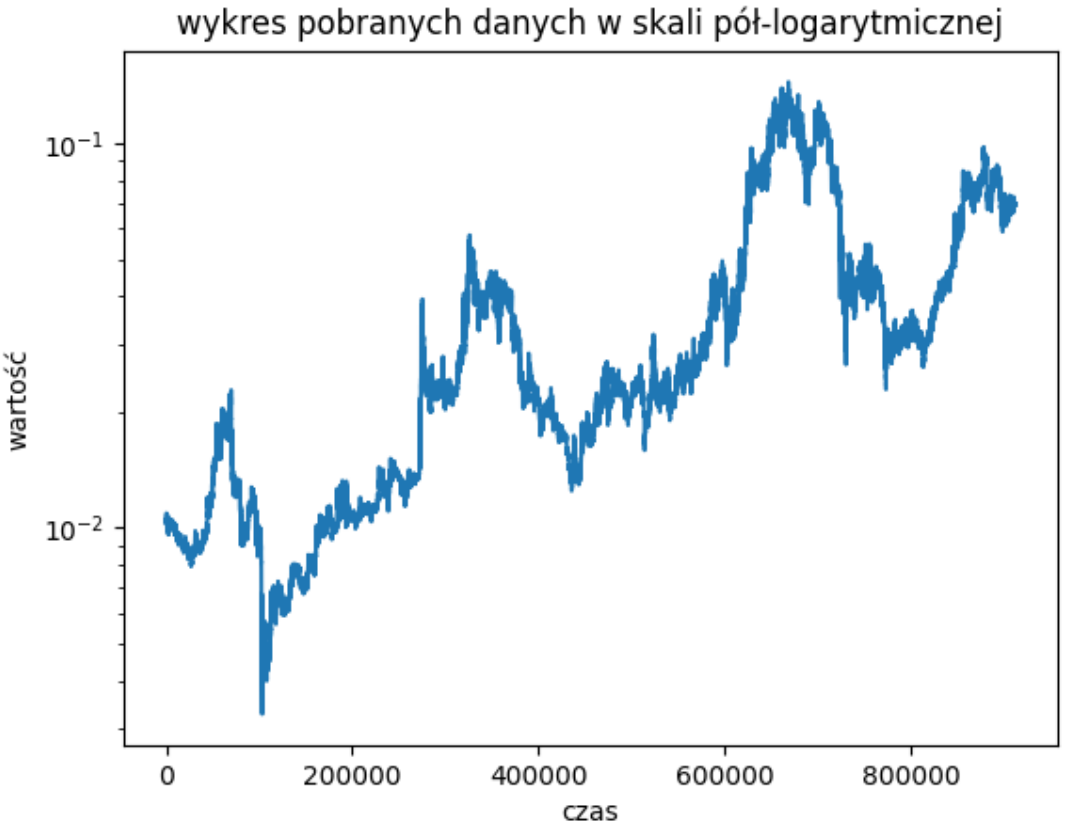
Dane pobrano w postaci pliku .csv i wczytano w następujący sposób.

```
dane = pd.read_csv(r'dane.csv', header=None)
```

Następnie wygenerowano wykresy danych w skali liniowej i pół-logarytmicznej.

```
def plot_plt(title, xlabel, ylabel, data, filename, scalexy=[True, True],
x_scale_log=False, y_scale_log=False):
    if x_scale_log:
        plt.xscale('log')
    if y_scale_log:
        plt.yscale('log')
    plt.title(title)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.plot(data, scalex=scalexy[0], scaley=scalexy[1])
    plt.savefig(filename)
    plt.close()
```

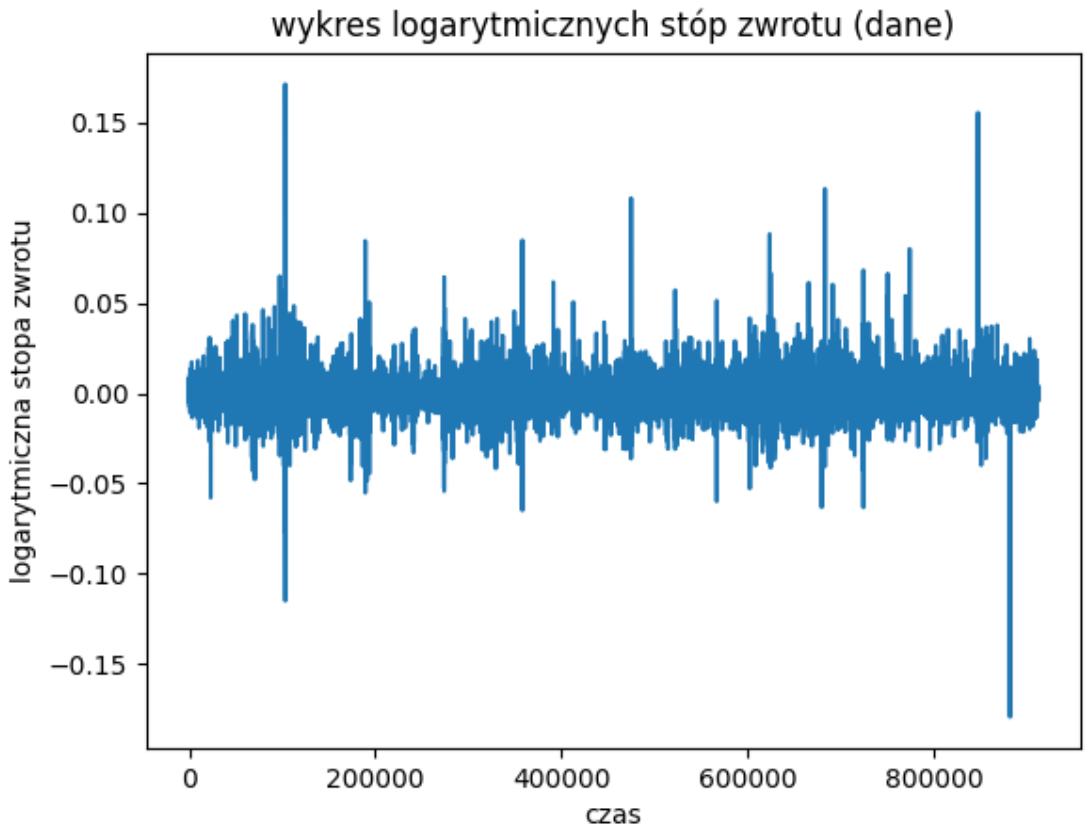

wykres pobranych danych w skali liniowej
wartość
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
200000
400000
600000
800000
czas


wykres pobranych danych w skali pół-logarytmicznej
wartość
10^{-1}
10^{-2}
0
200000
400000
600000
800000
czas

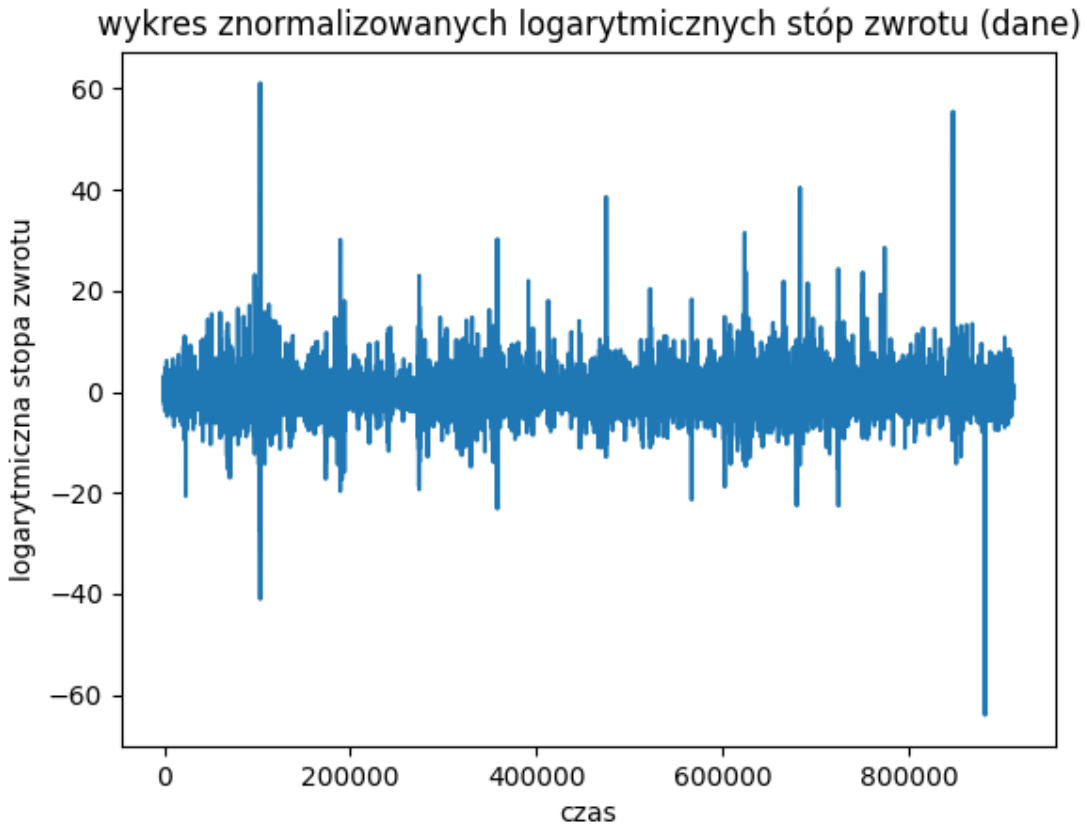
Obliczono logarytmiczne stopy zwrotu:

```
def zwroc_stopy_zwrotu(df):
    stopy_arr = []
    for i in range(len(df) - 1):
        stopy_arr.append((df[0][i + 1] - df[0][i]) / df[0][i])
    return stopy_arr
```

I wygenerowano wykres zarówno nieznormalizowanego szeregu stóp zwrotu :



Jak i znormalizowanego:



Sprawdzono również wartości średniej i odchylenia standardowego:

```
print("Srednia = ", round(stopyZwrotu_norm.mean()))
print("Odchylenie Standardowe = ", stopyZwrotu_norm.std())
```

Srednia = 0.0
Odchylenie Standardowe = 1.0

Temat 2.

Wygenerowano dane o rozkładzie normalnym (white noise):

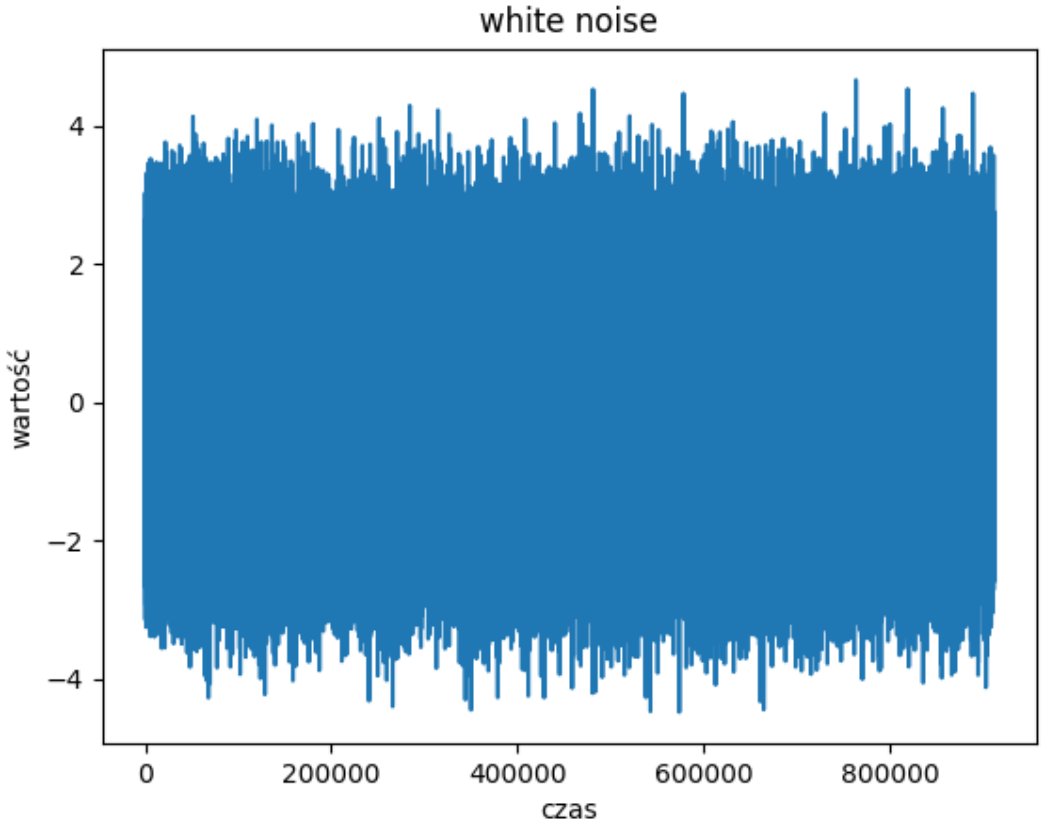
```
whiteNoise = np.random.normal(loc=0, scale=1, size=9076)
```

Sprawdzono wartości średniej i odchylenia standardowego:

```
print("Srednia = ", round(whiteNoise.mean()))
print("Odchylenie Standardowe = ", round(whiteNoise.std()))
```

Srednia = 0.0
Odchylenie Standardowe = 1.0

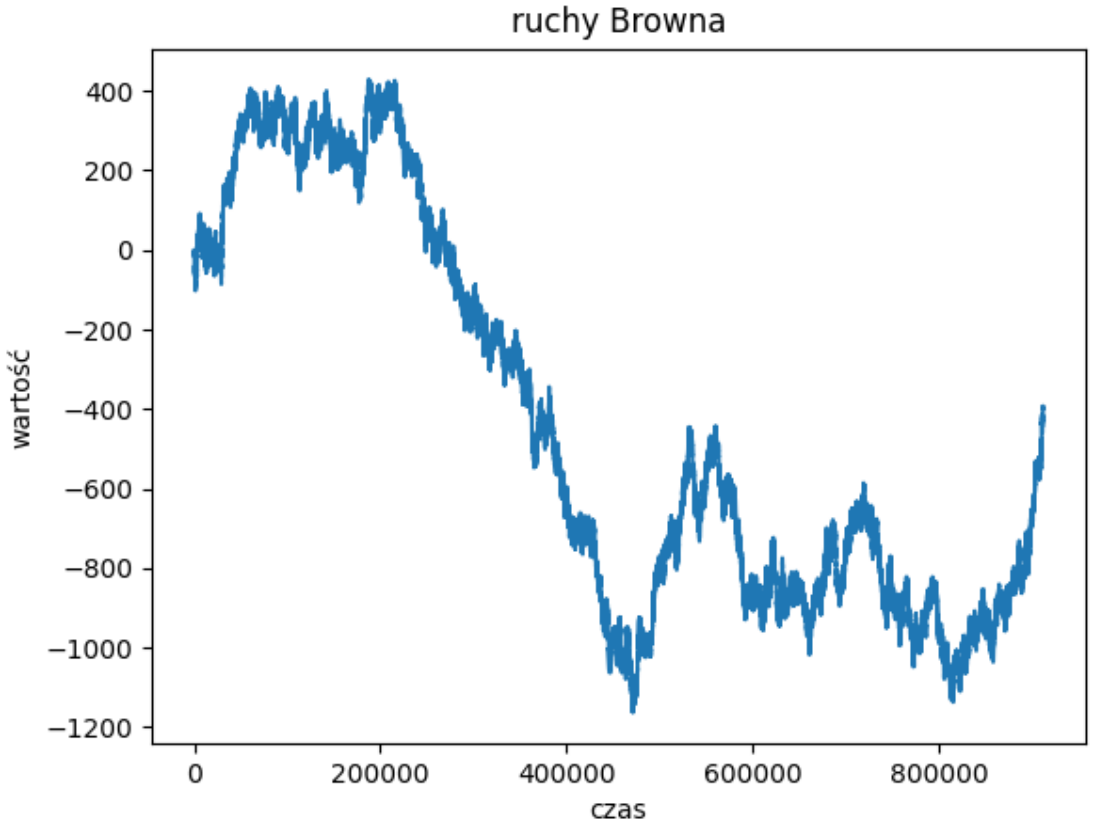
Następnie wygenerowano wykres danych:



Aby obliczyć sumę skumulowaną (ruchy Browna) zastosowano funkcję:

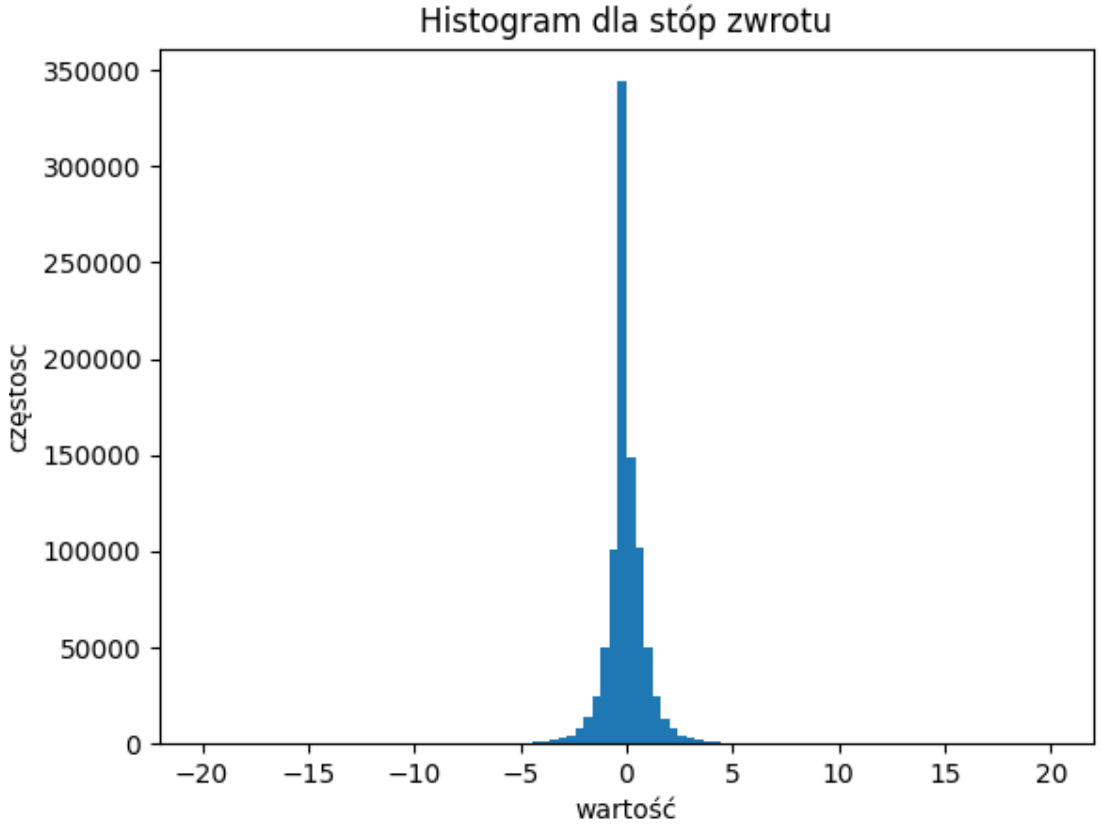
```
brown = np.cumsum(whiteNoise)
```

Wygenerowano wykres dla sumy skumulowanej:

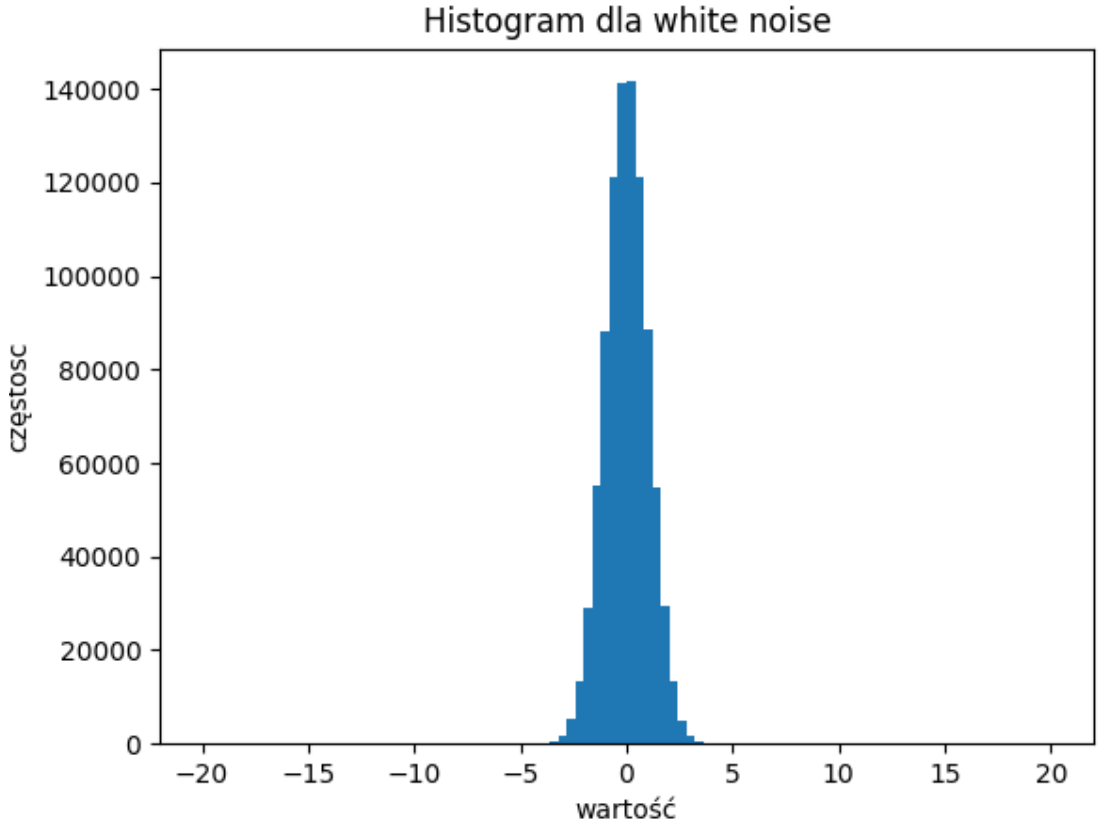


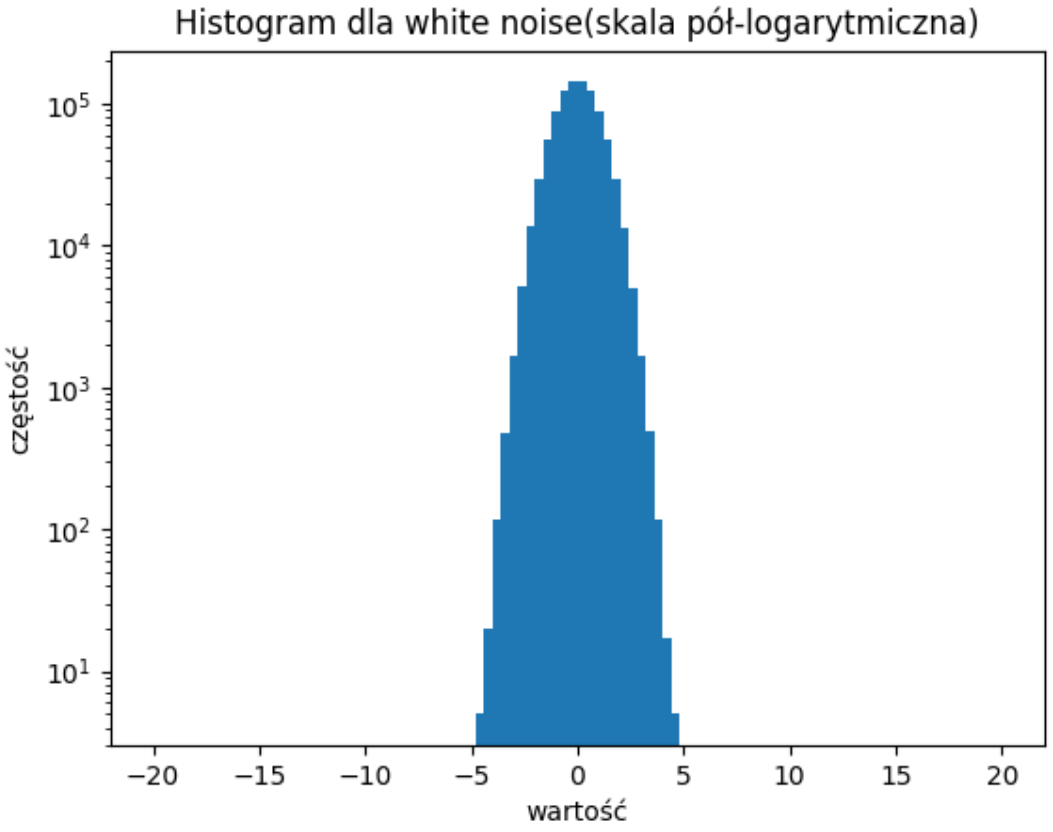
Temat 3.

Zarówno dla danych giełdowych jak i wygenerowanego szumu stworzono histogramy:

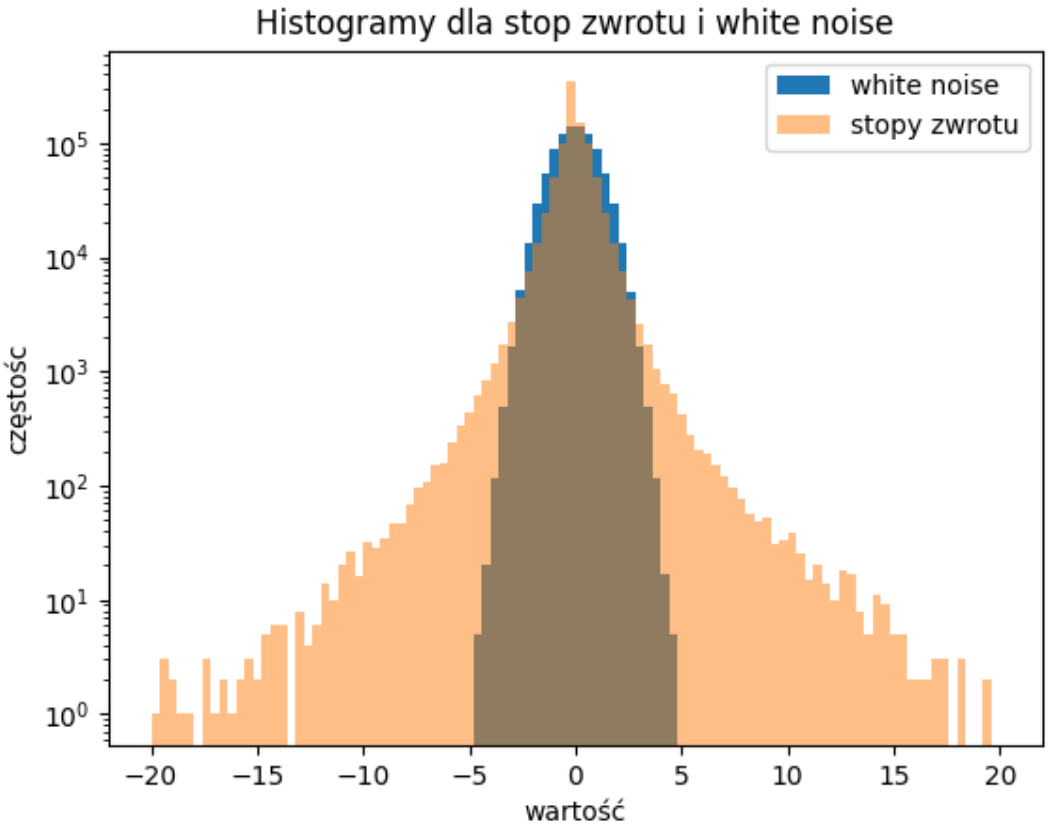

Histogram dla stóp zwrotu
częstosc
wartość
350000
300000
250000
200000
150000
100000
50000
0
−20
−15
−10
−5
0
5
10
15
20


Histogram dla stóp zwrotu(skala pół-logarytmiczna)
częstość
wartość
10^5
10^4
10^3
10^2
10^1
10^0
−20
−15
−10
−5
0
5
10
15
20


Histogram dla white noise
140000
120000
100000
80000
60000
40000
20000
0
częstosc
−20
−15
−10
−5
0
5
10
15
20
wartość

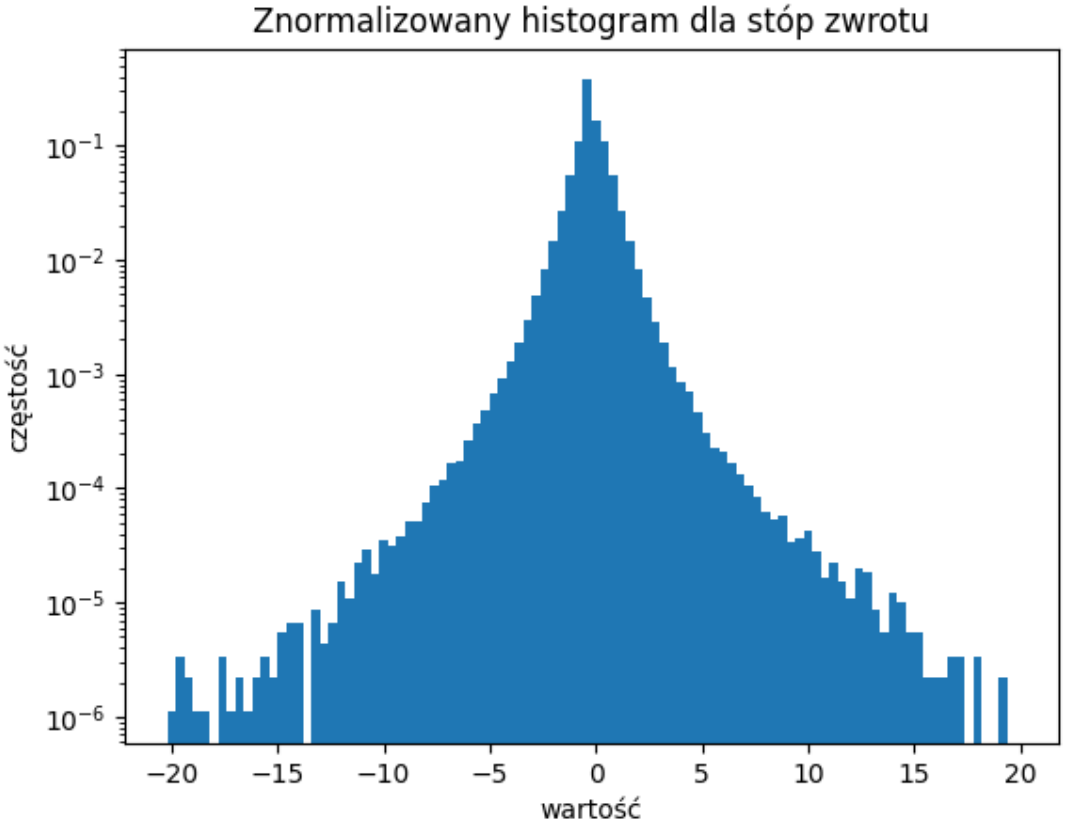

Histogram dla white noise(skala pół-logarytmiczna)
10^5
10^4
10^3
10^2
10^1
częstość
−20
−15
−10
−5
0
5
10
15
20
wartość

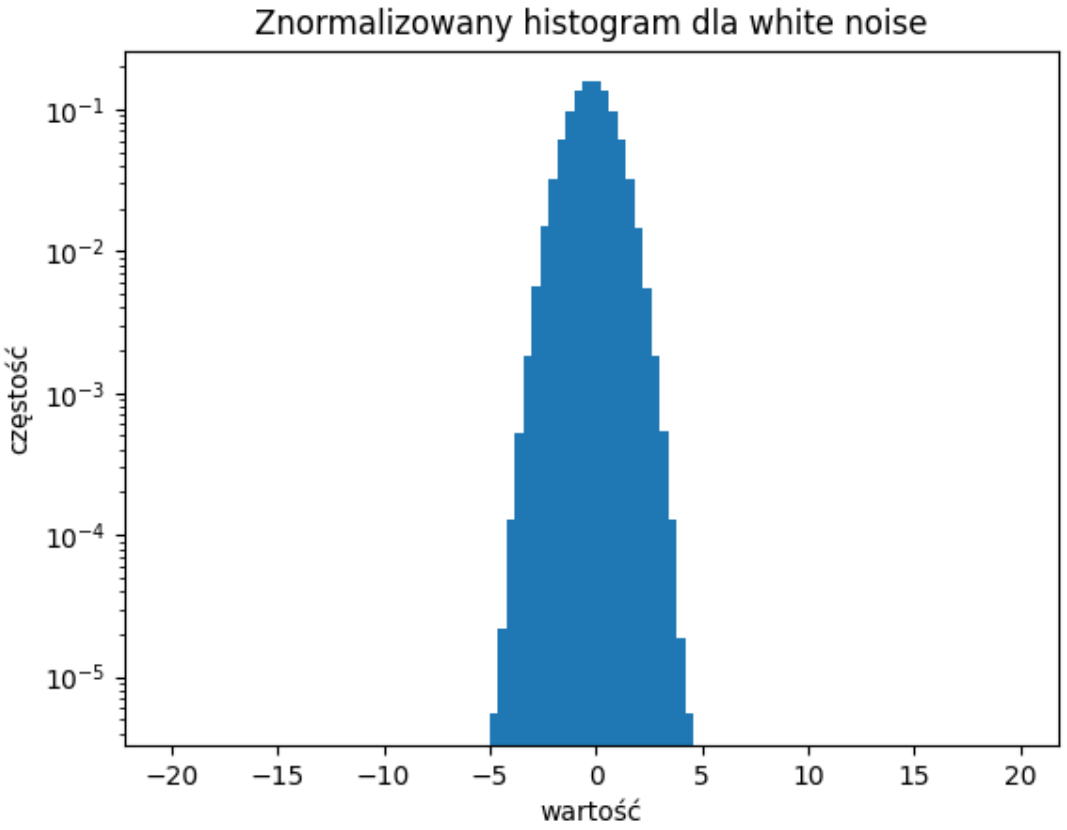
Przedstawiono je również na jednym wykresie w skali pół-logarytmicznej:



Następnie znormalizowano stworzone histogramy dzieląc wartość każdego słupka na sumę wartości wszystkich słupków. Wyrysowano znormalizowane histogramy:

```
StopySuma = sum(histStopyZwrotu[0])
WhiteSuma = sum(histWhiteNoise[0])
plt.plot(histStopyZwrotu[1][:-1], (histStopyZwrotu[0]/StopySuma))
plt.plot(histWhiteNoise[1][:-1], (histWhiteNoise[0]/WhiteSuma))
```


Znormalizowany histogram dla stóp zwrotu
częstość
wartość
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
−20
−15
−10
−5
0
5
10
15
20


Znormalizowany histogram dla white noise
częstość
wartość
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
−20
−15
−10
−5
0
5
10
15
20

Sprawdzono czy suma słupków jest równa 1:

```
print(sum(histWhiteNoise[0]/WhiteSuma))
print(sum(histWhiteNoise[0]/WhiteSuma))
```

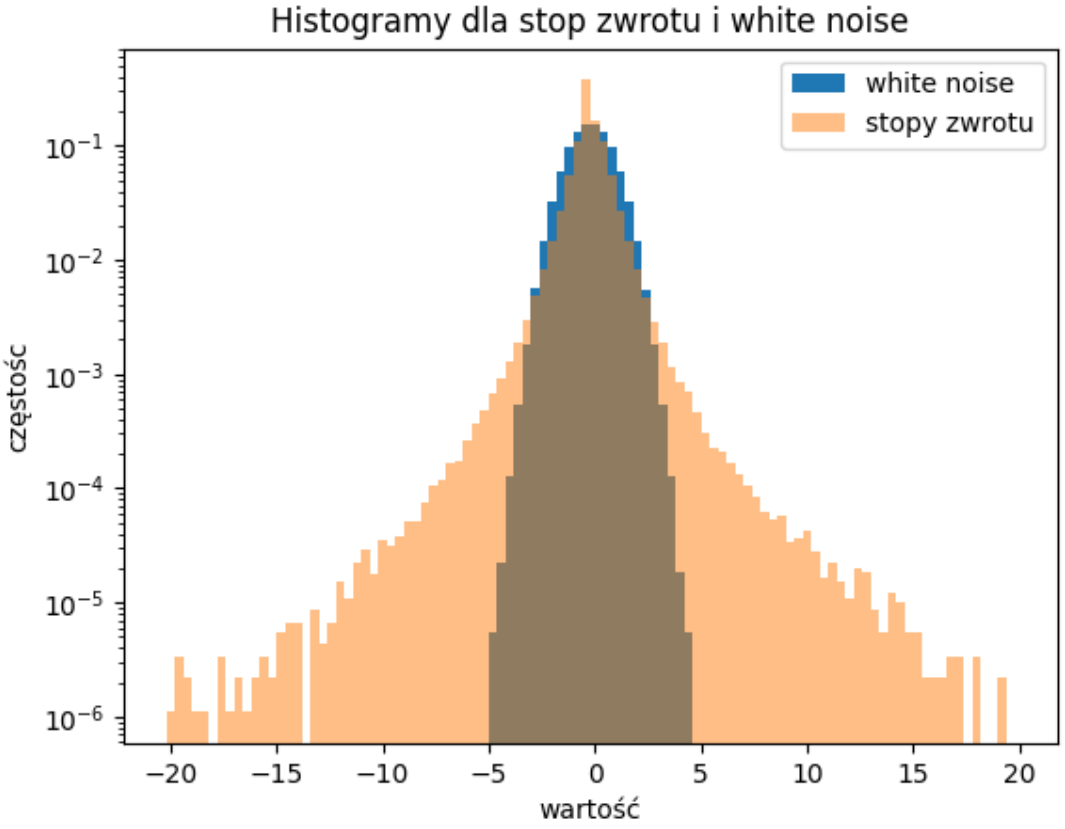
```
1.0
1.0
```

Aby policzyć skośność i kurtozę wykorzystano funkcje biblioteki scipy.stats.mstats:

```
print("Dane")
print("   Kurtoza: ", stat.kurtosis(stopyZwrotu_norm))
print("   Skośność: ", stat.skew(stopyZwrotu_norm))

print("White Noise")
print("   Kurtoza: ", stat.kurtosis(whiteNoise))
print("   Skośność: ", stat.skew(whiteNoise))
```

```
Dane giełdowe
   Kurtoza:  [98.97735718]
   Skośność:  [0.66026077]
White Noise
   Kurtoza:  0.004388238394612998
   Skośność:  0.000714506761190646
```

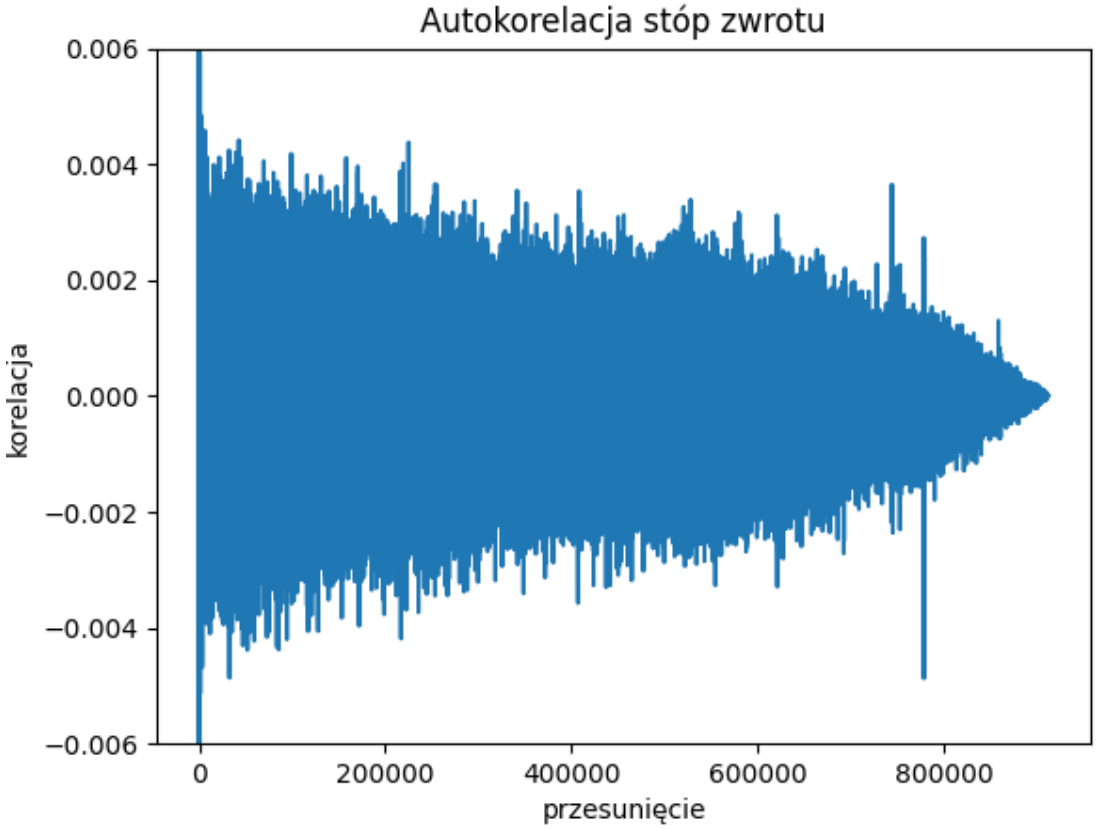
Przedstawiono oba znormalizowane rozkłady na jednym wykresie:

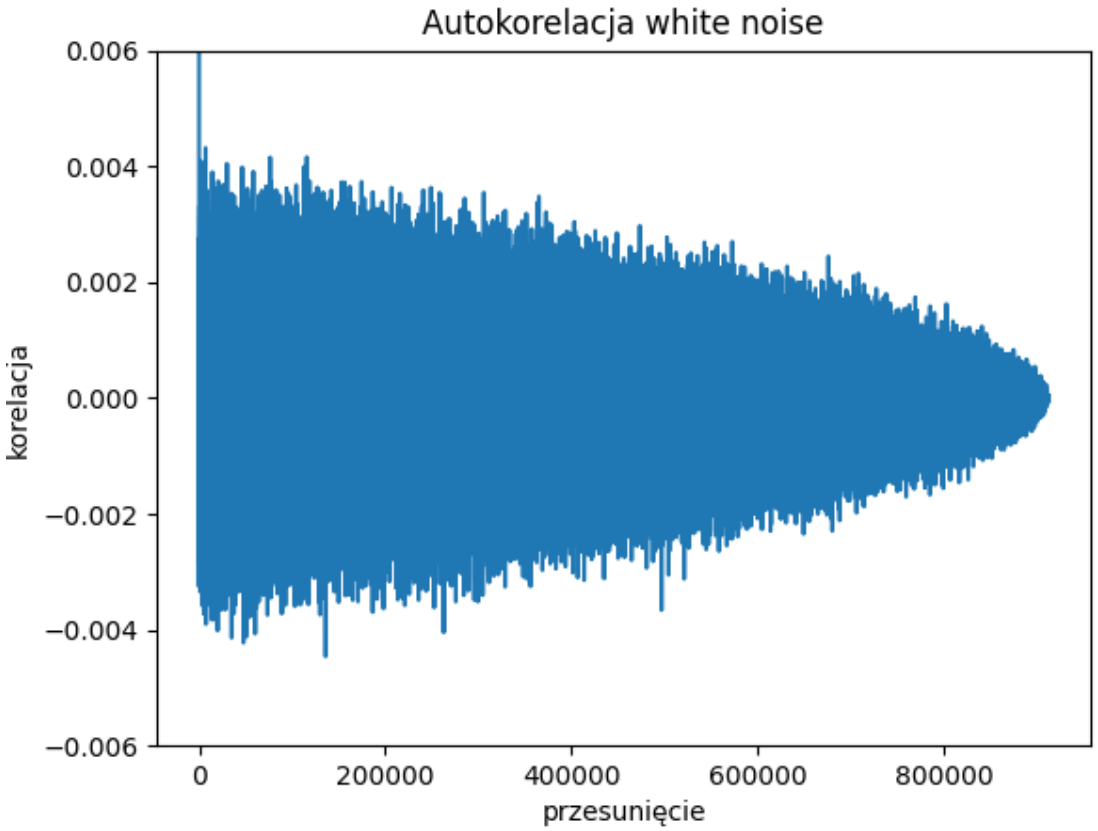


Temat 4.

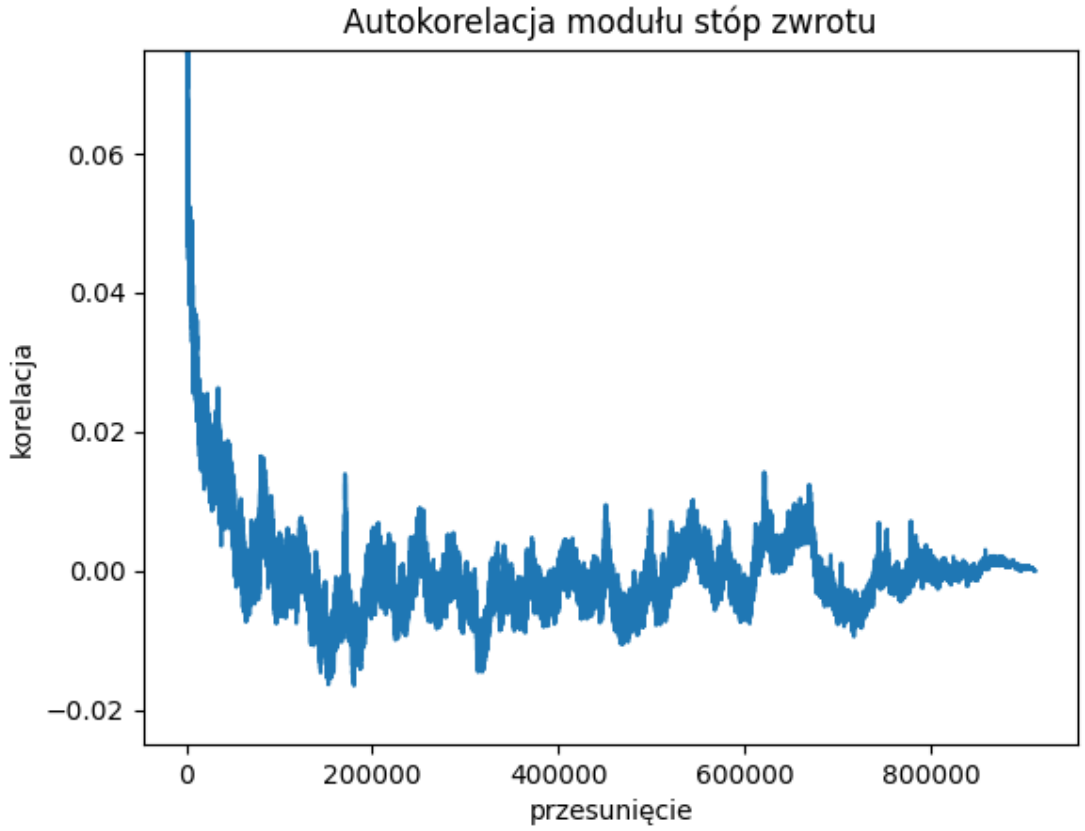
Zarówno dla danych giełdowych jak i wygenerowanych policzono funkcję autokorelacji.

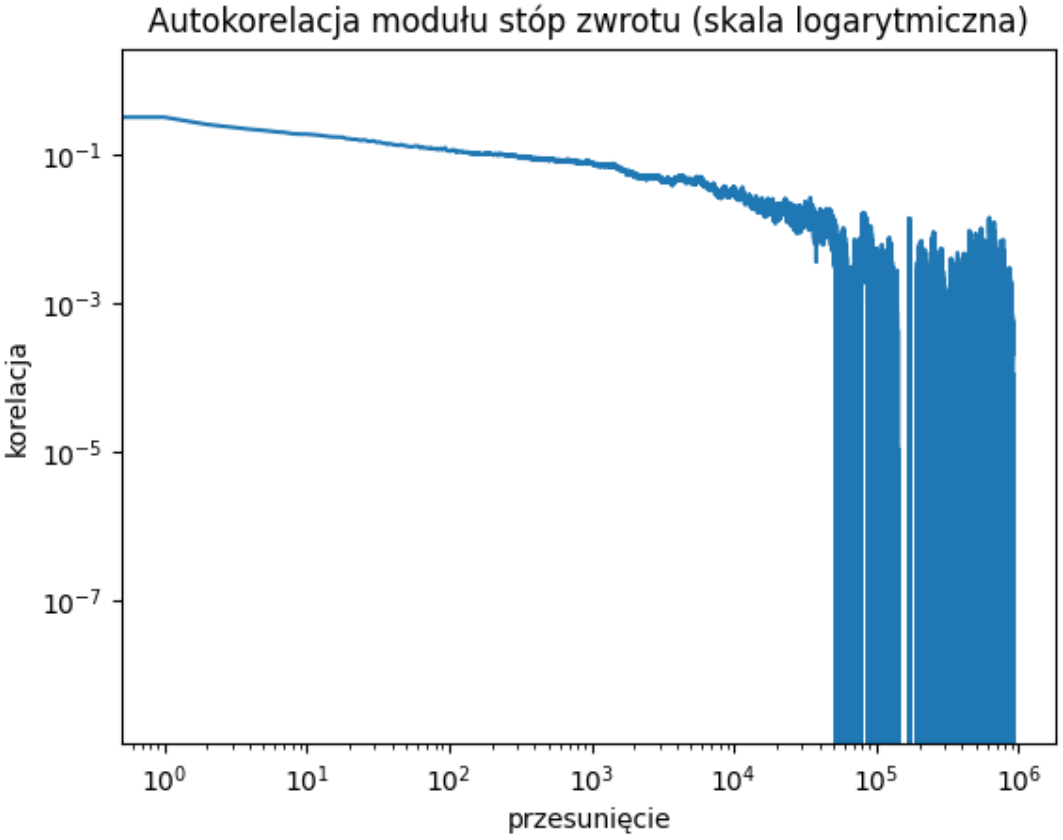
```
acf_stopy = stats.tsa.acf(stopyZwrotu_norm, nlags=len(stopyZwrotu_norm) - 1)
# autokorelacja stop zwrotu

acf_stopy_abs = stats.tsa.acf(abs_stopy, nlags=len(stopyZwrotu_norm) - 1)  #
autokorelacja modulu stop zwrotu

acf_white_noise = stats.tsa.acf(whiteNoise, nlags=len(whiteNoise) - 1)
# autokorelacja bialego szumu

acf_white_noise_abs = stats.tsa.acf(abs_white_noise,
nlags=len(abs_white_noise) - 1)
# autokorelacja modulu bialego szumu
```

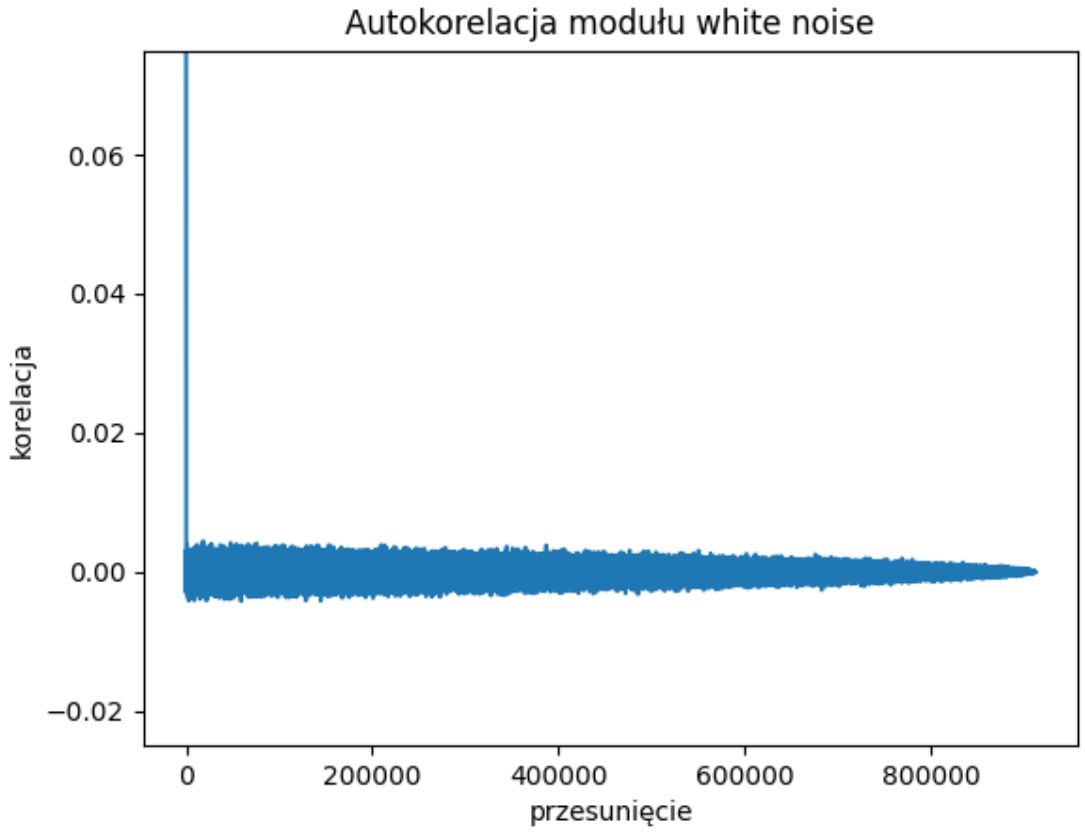

Autokorelacja stóp zwrotu
0.006
0.004
0.002
0.000
−0.002
−0.004
−0.006
korelacja
0
200000
400000
600000
800000
przesunięcie

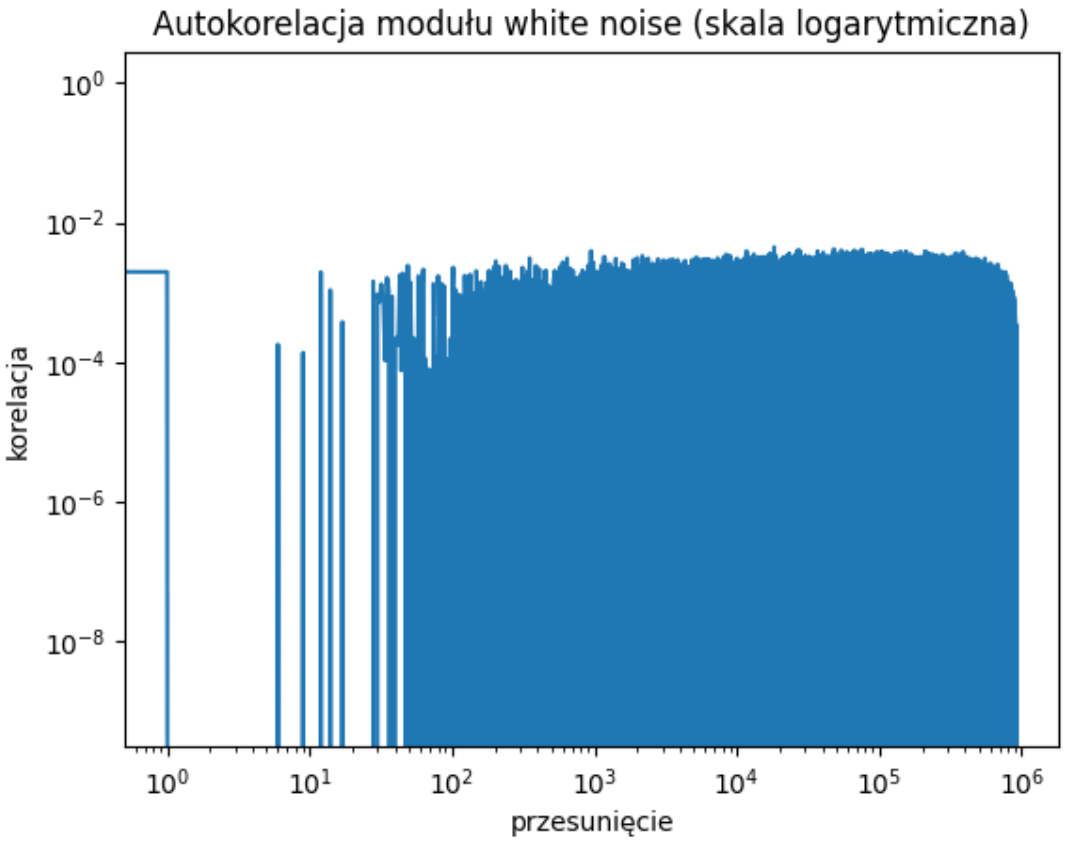

Autokorelacja white noise
0.006
0.004
0.002
0.000
−0.002
−0.004
−0.006
korelacja
0
200000
400000
600000
800000
przesunięcie

Dla modułów danych giełdowych jak i wygenerowanych również policzono funkcję autokorelacji.




Autokorelacja modułu stóp zwrotu (skala logarytmiczna)
korelacja
10^{-1}
10^{-3}
10^{-5}
10^{-7}
10^{0}
10^{1}
10^{2}
10^{3}
10^{4}
10^{5}
10^{6}
przesunięcie


Autokorelacja modułu white noise
korelacja
0.06
0.04
0.02
0.00
−0.02
0
200000
400000
600000
800000
przesunięcie

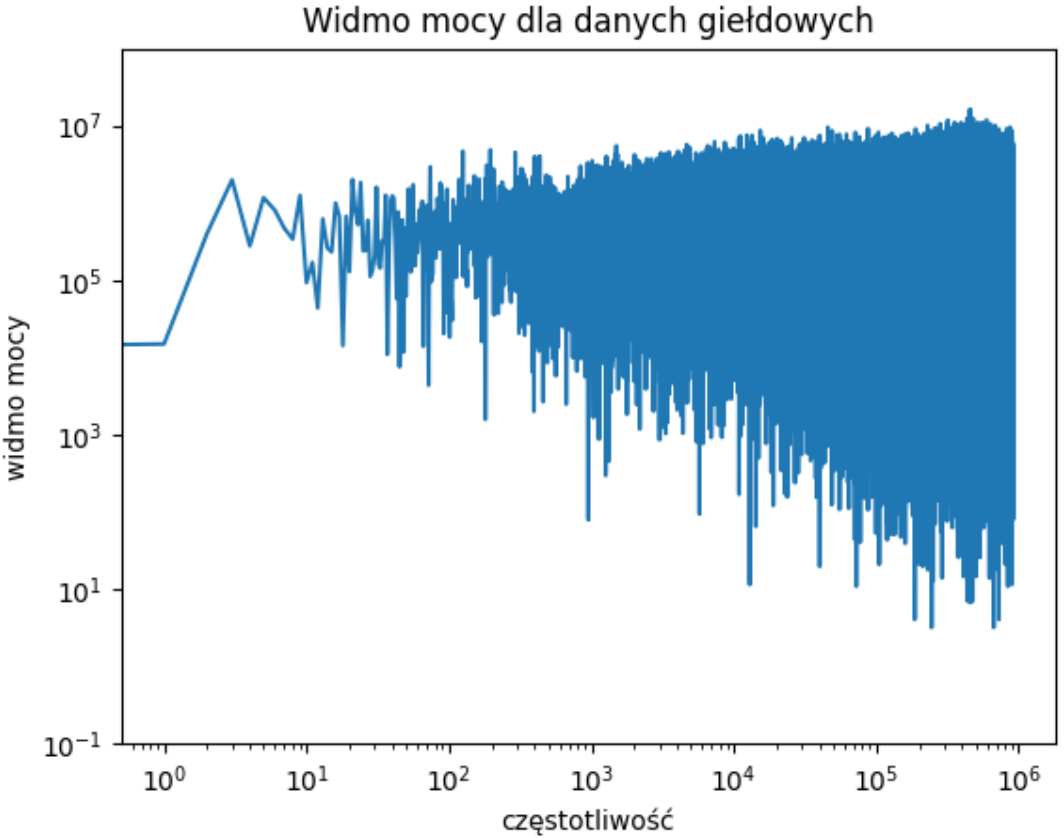


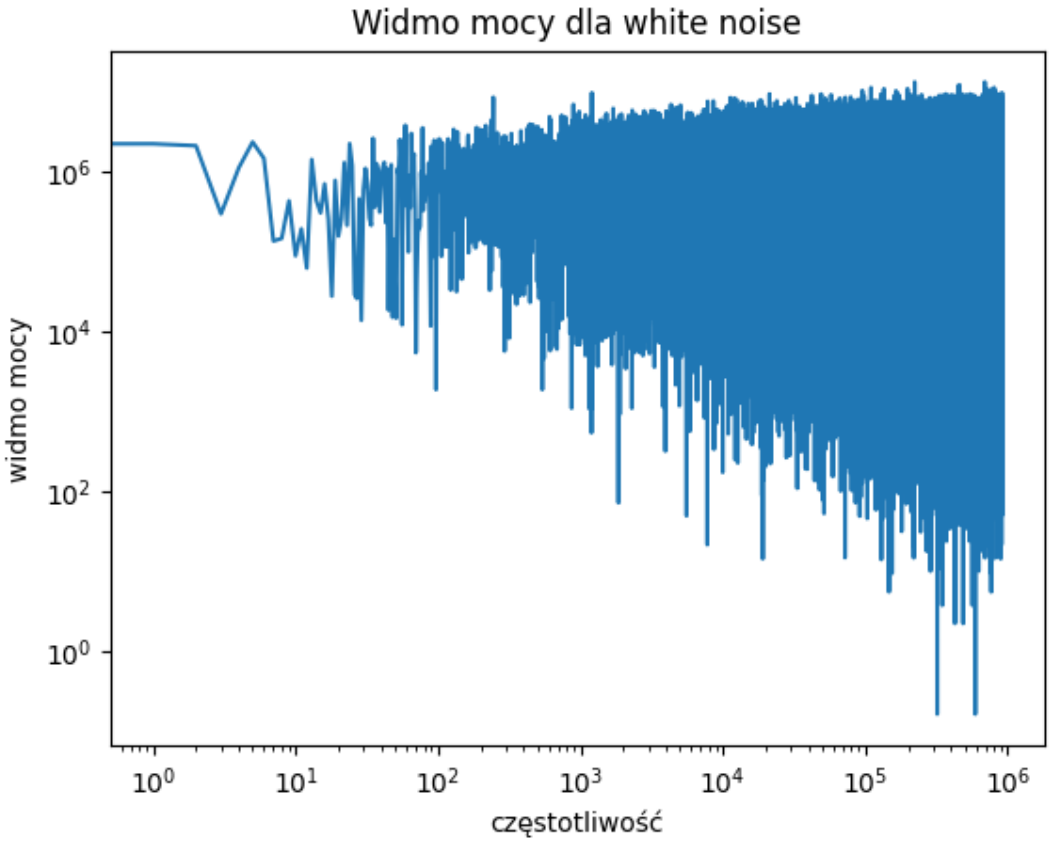
Następnie stworzono wykresy dla widma mocy sygnałów (skala logarytmiczna)

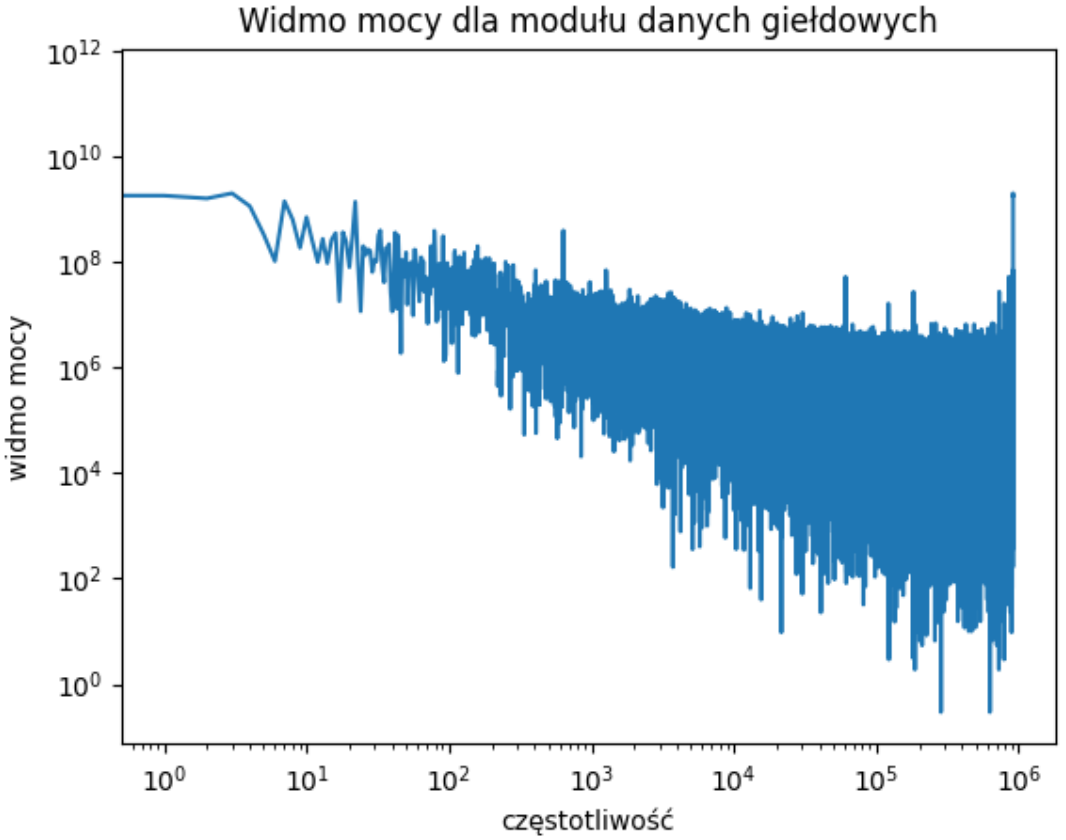
```
power_spectrum_stopy_zwrotu = np.abs(np.fft.fft(stopyZwrotu_norm)) ** 2
power_spectrum_white_noise = np.abs(np.fft.fft(whiteNoise)) ** 2
```

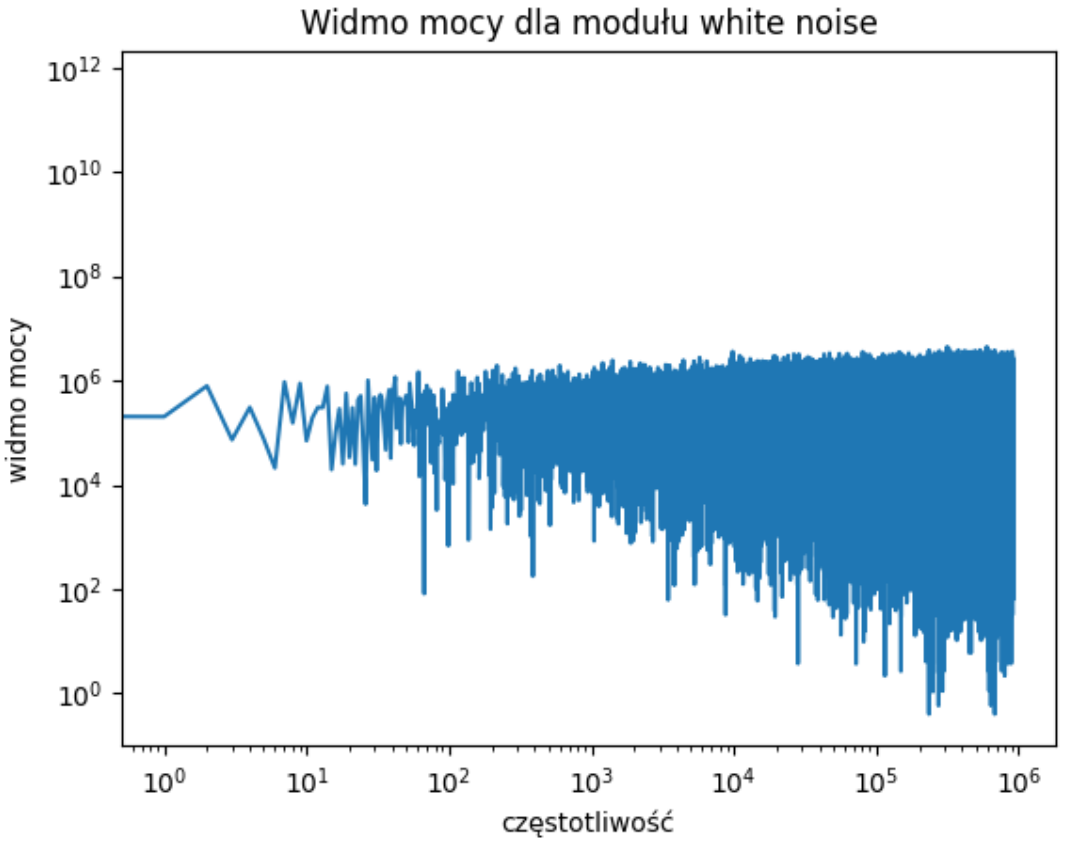
oraz dla widma mocy modułów sygnałów (również w skali logarytmicznej).

```
power_spectrum_stopy_zwrotu_abs = np.abs(np.fft.fft(abs(stopyZwrotu_norm))) ** 2
power_spectrum_white_noise_abs = np.abs(np.fft.fft(abs_white_noise)) ** 2
```


Widmo mocy dla danych giełdowych
widmo mocy
częstotliwość


Widmo mocy dla white noise
widmo mocy
częstotliwość


Widmo mocy dla modułu danych giełdowych
widmo mocy
częstotliwość
10^{12}
10^{10}
10^{8}
10^{6}
10^{4}
10^{2}
10^{0}
10^{0} 10^{1} 10^{2} 10^{3} 10^{4} 10^{5} 10^{6}


Widmo mocy dla modułu white noise
widmo mocy
częstotliwość
10^{12}
10^{10}
10^{8}
10^{6}
10^{4}
10^{2}
10^{0}
10^{0} 10^{1} 10^{2} 10^{3} 10^{4} 10^{5} 10^{6}