IONIAN UNIVERSITY
DEPARTMENT OF INFORMATICS

STOCHASTIC DATA ANALYSIS

Semester Assignment

Markou Dimitra

Markov Stochastic Process

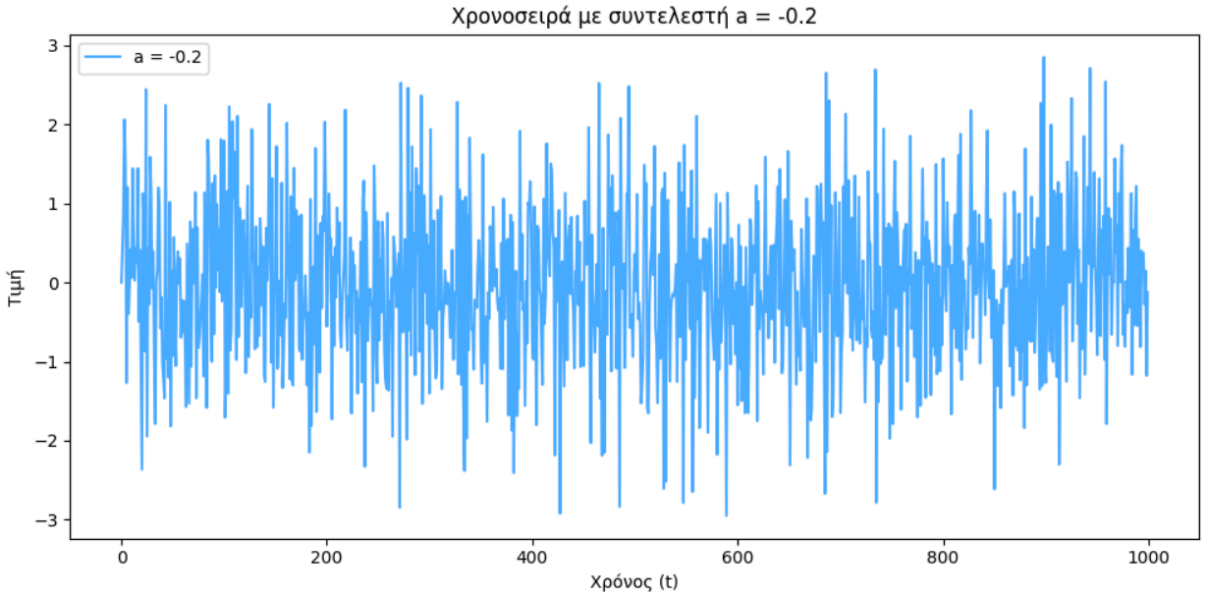
$$X_t = a \cdot X_{t-1} + Z_t$$

The above definition represents a first-order autoregressive process (AR(1)). It implies that the current value of the series X_t is generated by multiplying the previous value X_{t-1} by the coefficient a and then adding a white noise error term Z_t. The white noise term Z_t introduces randomness, ensuring that the process is stochastic rather than deterministic.

- X_t is the value of the time series at time t.
- a is the autoregressive coefficient, which determines the effect of the previous value X_{t-1} on the current value X_t.
- Z_t rrent value.
- is the white noise term at the time t, which is a random error term that is assumed to be normally distributed with a mean of 0 and a constant variance.

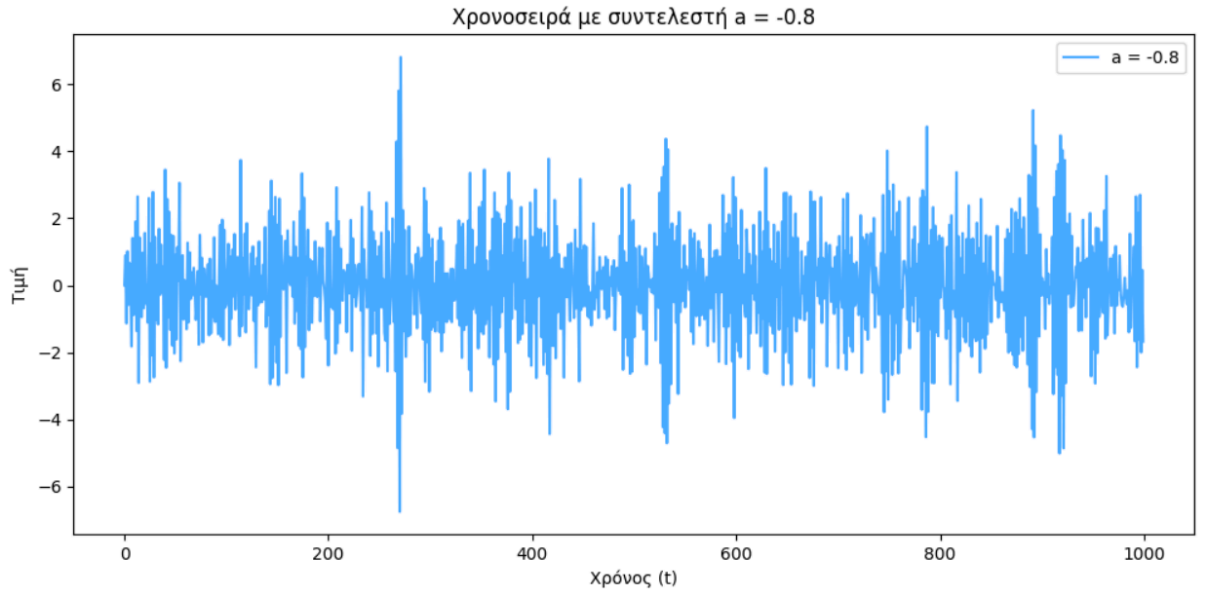
Time series with $a = -0,2$:

The time series has a moderate negative autoregressive coefficient, meaning that each value is affected by about 20% of the previous value in the opposite direction, plus some random noise.



Time series with $a = -0,8$:

The time series has a stronger negative autoregressive coefficient , meaning that each value is affected by about 80% of the previous value in the opposite direction, plus some random noise. This higher coefficient results in more pronounced fluctuations compared to the first series.



Markov Stochastic Process: Code

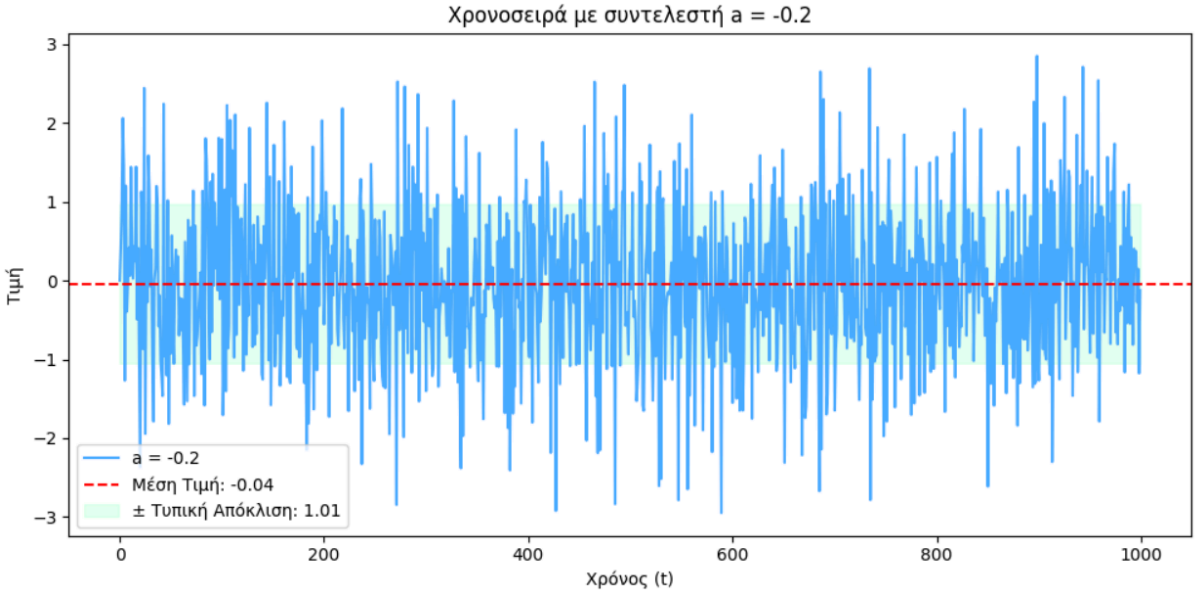
```
import numpy as np

# Παράμετροι
n = 1000  # Μήκος Χρονοσειράς
a1 = -0.2
a2 = -0.8
np.random.seed(0)

# Δημιουργία Λευκού Θορύβου
Zt1 = np.random.normal(0, 1, n)
Zt2 = np.random.normal(0, 1, n)

# Αρχικοποίηση Χρονοσειρών
X1 = np.zeros(n)
X2 = np.zeros(n)

for t in range(1, n):
     X1[t] = a1 * X1[t-1] + Zt1[t]
     X2[t] = a2 * X2[t-1] + Zt2[t]
```

The above code first introduces the NumPy library, then sets the parameter $n = 1000$, which denotes the length of the time series, as well as the coefficients $a1 = -0.2$ and $a2 = -0.8$. The `np.random.seed(0)` sets the seed for NumPy's random number generator to ensure repeatability of the generated random numbers. To generate white noise we use `np.random.normal(0, 1, n)`. Then we initialize the time series with `np.zeros(n)` for length ν . The iteration loop multiplies the previous value of each time series by the coefficient and adds the white noise.

Mean Value, Variance and Standard Deviation

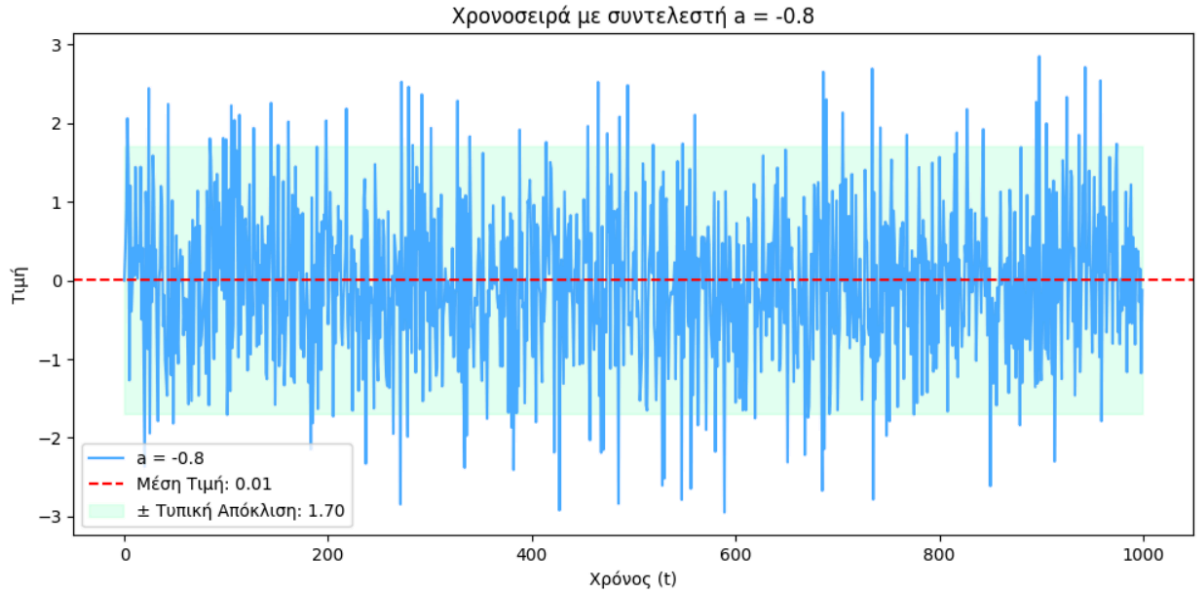
Time series with $a = -0.2$:

- **Mean Value:** -0.039204407048413166
- **Variance:** 1.0226585877909087
- **Standard Deviation:** 1.0112658343832786



Time series with $a = -0.8$:

- **Mean Value:** 0.006509721478170432
- **Variance:** 2.8934121695352677
- **Standard Deviation:** 1.7010032832229536



Mean Value, Variance and Standard Deviation: Interpretation

Time series with $a = -0.2$:

- **Mean Value:** -0.039204407048413166

The mean value of the time series is close to zero, this suggests that on average, the values of the time series fluctuate around zero. This is expected for a time series with a small negative coefficient, as the effect of past values on the current value is relatively weak.

- **Variance:** 1.0226585877909087

Variance measures the dispersion of the values of the time series from the mean. A variance of around 1 indicates moderate variability of the data. The values do not have a large dispersion from the mean.

- **Standard Deviation:** 1.0112658343832786

The standard deviation also indicates moderate variability. It provides a measure of the average distance of the time series values from the mean. A standard deviation of about 1 is consistent with observed variability.

Time series with $a = -0.8$:

- **Mean Value:** 0.006509721478170432

The mean value is very close to zero, indicating that the values of the time series average around zero. This is typical for a time series, where the strong negative coefficient ensures that large deviations in one direction are followed by significant corrections in the opposite direction.

- **Variance:** 2.8934121695352677

The variance is significantly higher than that of the first time series. This suggests a higher degree of variability in the data, which is expected due to the stronger negative coefficient. The values tend to vary more from the mean compared to the first series.

- **Standard Deviation:** 1.7010032832229536

The standard deviation is also higher, reflecting the increased volatility of the time series. A standard deviation of 1.7 means that the values in the series are on average 1.7 points away from the mean, indicating more significant variations.

The time series with $a = -0,2$ shows moderate volatility around the mean, with values generally remaining closer to the mean, while the time series with $a = -0,8$ is more volatile, with values further away from the mean, leading to more pronounced fluctuations. The difference in coefficients a affects the degree of dependence on previous values, thus affecting the volatility and stability of the time series. A higher magnitude of the coefficient leads to greater volatility and larger corrections after deviations, as shown in the second row.

Mean Value, Variance and Standard Deviation: Code

```
# Υπολογισμός Μέσης Τιμής, Διακύμανσης και Τυπικής Απόκλισης

mean_X1 = np.mean(X1)
variance_X1 = np.var(X1)
std_dev_X1 = np.std(X1)
print("Για την χρονοσειρά με συντελεστή a = -0.2:")
print("Μέση Τιμή (Mean):", mean_X1)
print("Διακύμανση (Variance):", variance_X1)
print("Τυπική Απόκλιση (Standard Deviation):", std_dev_X1, "\n")

mean_X2 = np.mean(X2)
variance_X2 = np.var(X2)
std_dev_X2 = np.std(X2)
print("Για την χρονοσειρά με συντελεστή a = -0.8")
print("Μέση Τιμή (Mean):", mean_X2)
print("Διακύμανση (Variance):", variance_X2)
print("Τυπική Απόκλιση (Standard Deviation):", std_dev_X2)
```

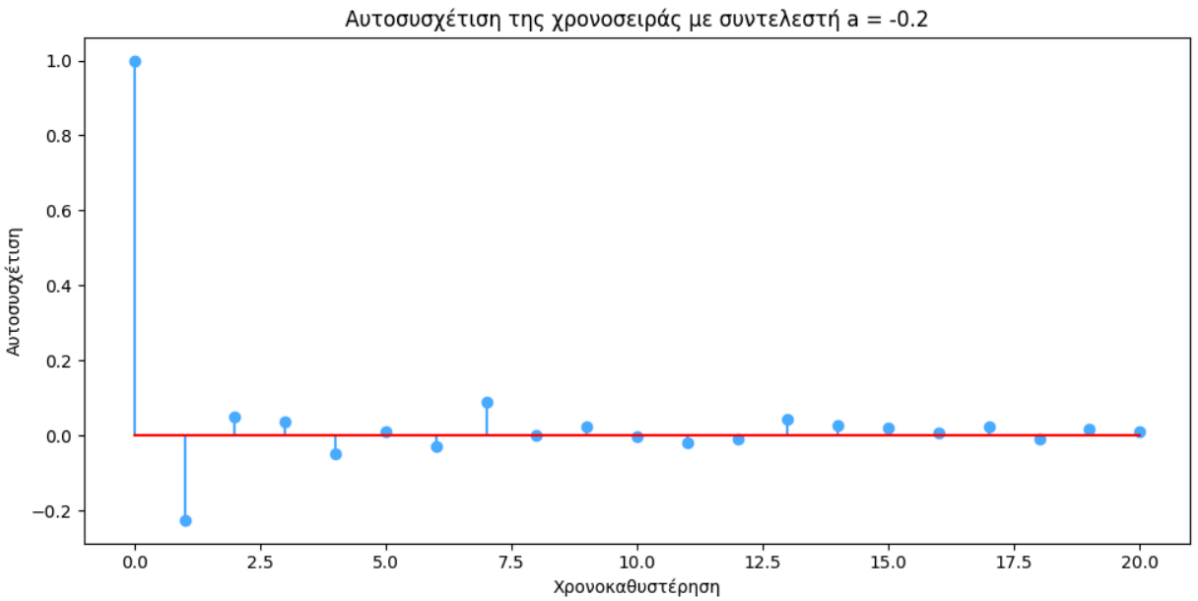
The above code calculates the mean value, variance and standard deviation with the NumPy library.

Autocorrelation

The autocorrelation function provides information about the "memory" of a time series.

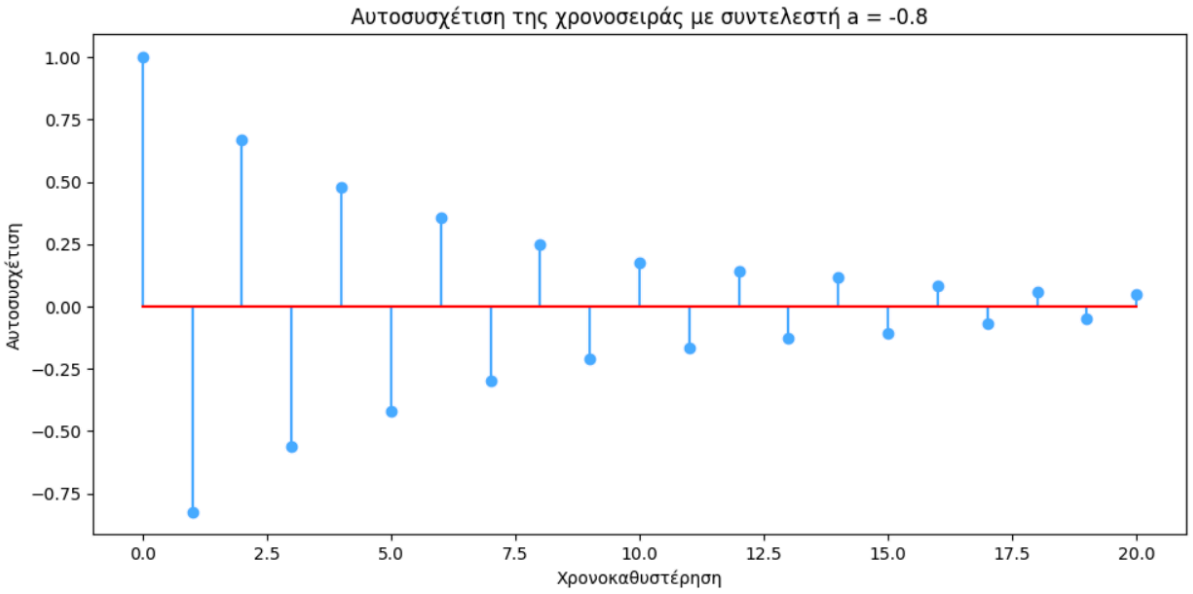
Time series with $a = -0.2$:

The autocorrelation function shows a relatively fast decay. At the moment when the time delay equals 1, a negative autocorrelation is observed, which is expected since the coefficient $a = -0.2$ introduces a mild negative dependence. For subsequent time lags, the autocorrelation values will decrease rapidly, approaching zero with little oscillation around it, indicating that the influence of the previous values is decreasing rapidly.



Time series with $a = -0.8$:

The autocorrelation function shows a slower decay, reflecting a stronger dependence on past values. At the time when the time lag equals 1, we observe a large negative autocorrelation, due to the strong negative feedback introduced by the coefficient $a = -0,8$. For subsequent time lags the autocorrelation values will show an oscillating pattern around zero, gradually approaching it, reflecting the alternating influence of past values but decaying more slowly compared to the first time series.



Autocorrelation: Interpretation

In general, forecasts based on recent values are more accurate, but the time series is more prone to random fluctuations, while time series with long memory may be more predictable over longer horizons, but strong negative correlations may lead to more pronounced oscillations.

The time series with a coefficient $a = -0.2$ shows a rapidly decreasing flow, indicating that the influence of the previous values is decreasing rapidly. This suggests that the time series has a short memory, meaning that the current value is only weakly dependent on the values of the distant past.

The time series with a coefficient $a = -0.8$ shows a slower decline compared to the previous time series, indicating that the influence of past values persists longer. This suggests that the time series has a longer memory, meaning that the current value is more dependent on past values.

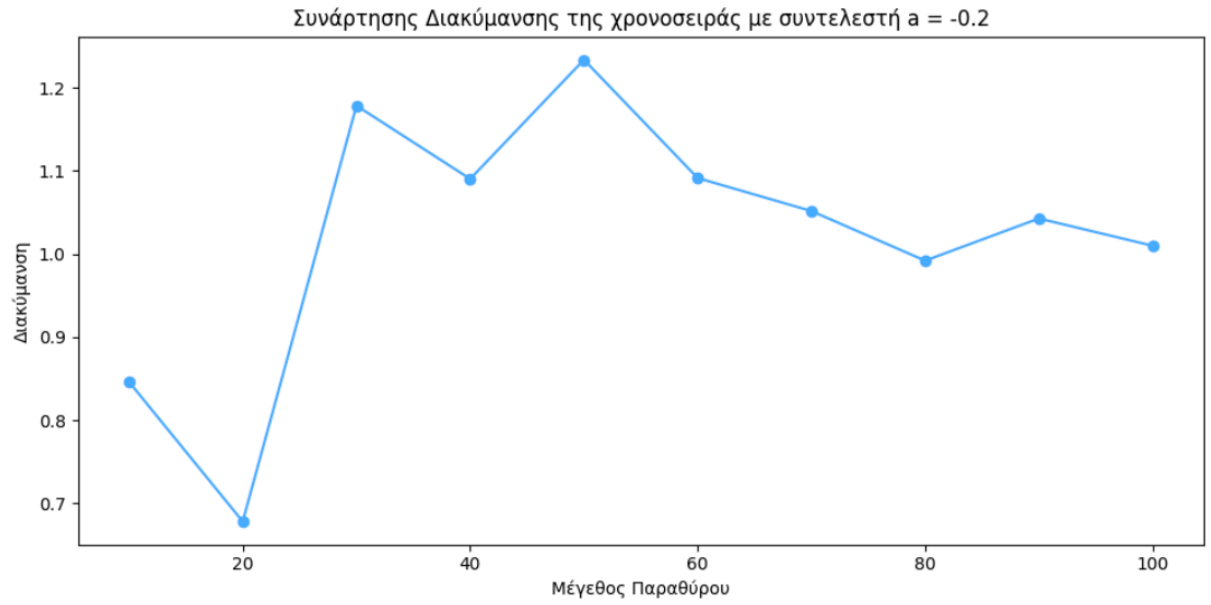
Autocorrelation: Code

```
# Υπολογισμός αυτοσυχέτισης - autocorrelation
acf_X1 = sm.tsa.acf(X1, nlags=20)
acf_X2 = sm.tsa.acf(X2, nlags=20)
```

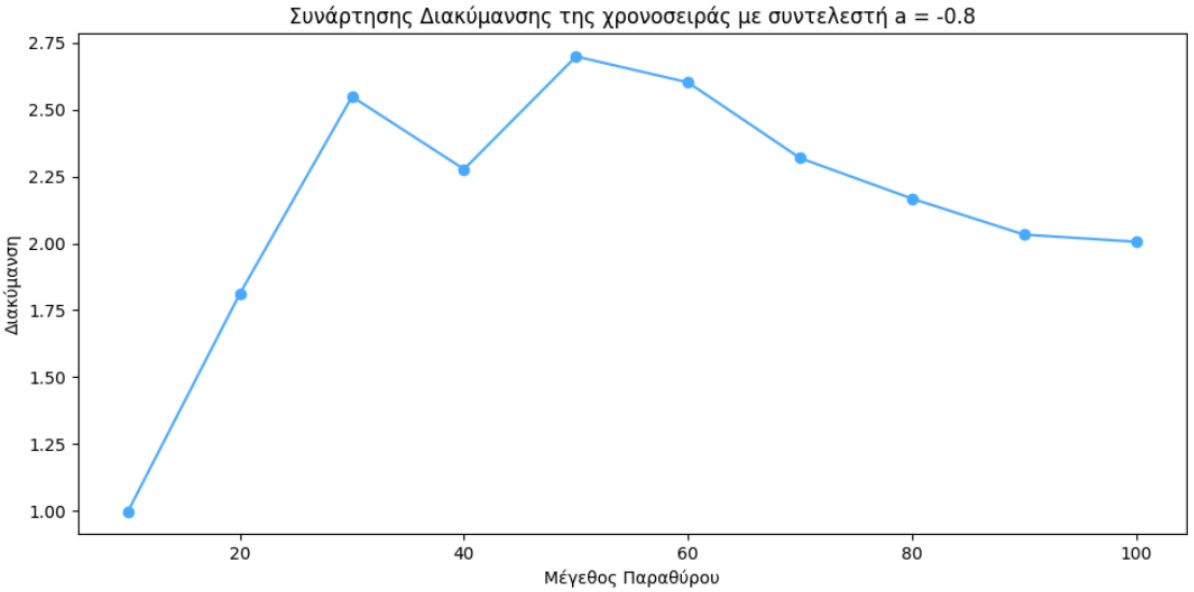
For the calculation of autocorrelation, the `statsmodels.api` was used. I tried to use the function `curve_fit` from the `scipy.optimize`, library, unfortunately without success.

Variance Function

Time series with $a = -0.2$:



Time series with $a = -0.8$:



Variance Function: Interpretation

The variance function measures how the variability of the time series varies as a function of the size of the window. By looking at the variance at different window sizes, we understand how the time series behaves in short and long periods.

For a time series with short memory, the variance remains relatively constant with increasing window size. This suggests that the influence of the previous values decreases rapidly and the time series behaves more like white noise. For the time series with $a = -0,2$ the variance does not increase significantly with window size, indicating less memory and less persistent variability.

When the time series has a large memory, the variance increases with the size of the window. This is because the cumulative effect of previous values continues to affect the series as the window size increases. For the time series with a $a = -0,8$ coefficient of , the variance increases more markedly with window size, indicating greater memory and more persistent variability.

Variance Function: Code

```
# Υπολογισμός Συνάρτησης Διακύμανσης
window_sizes = np.arange(10, 101, 10)
rolling_variance_X1 = [np.var(X1[:w]) for w in window_sizes]
rolling_variance_X2 = [np.var(X2[:w]) for w in window_sizes]
```

The above code uses the NumPy library function `np.arange(10, 101, 10)` This function produces a table of window sizes starting from 10 to 100 (including), in increments of 10.

Project Conclusion

In this work, analysis was performed on two stochastic processes modeled by autoregressive time series with different coefficients, $a1 = -0,2$ and $a2 = -0,8$. Through the examination of statistical metrics, autocorrelation functions and variance function analyses, we demonstrated how the autoregressive coefficient significantly affects the behavior and characteristics of a time series. A smaller coefficient leads to mild fluctuations and lower volatility, indicating short memory, while a larger coefficient produces more pronounced fluctuations and higher volatility, reflecting longer memory and greater persistence. These findings underscore the importance of understanding the fundamental parameters of stochastic processes, which can have profound implications for various real-world applications, including financial markets, weather forecasting, and economic modeling.

This work has provided me with valuable insights into the field of stochastic process analysis and paves the way for future research and involvement in the field of statistics at both academic and professional levels.

The code of this thesis has been written in the Python programming language, and in the kaggle environment, where the code is available for viewing.

The libraries used are the following:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
```

Kaggle Link for the Project:
https://www.kaggle.com/code/markedd/stohastic-data-analysis-semester-project