Serie 13

Aufgabe 13.1

Es sei $\{X_1, X_2, \dots\}$ ein diskreter stochastischer Prozess. Zeigen Sie, dass

$$\gamma(i, j) = \mathrm{E}(X_i X_j) - \mu(i)\mu(j).$$

Aufgabe 13.2

In dieser Aufgabe werden wir die empirische Autokorrelationsfunktion diverser Zeitreihen berechnen und graphisch darstellen. Insbesondere interessiert uns die Autokorrelationsfunktion verrauschter Signale.

a) Berechnen Sie die Autokorrelationsfunktion des Börsenkurses von Tesla und stellen Sie diese graphisch dar. **Python** -Hinweis:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from pandas import DataFrame
from statsmodels.graphics.tsaplots import plot_acf
Tesla = pd.read_csv(".../Tesla.csv", sep="\t",header=0)

Tesla["Date"] = pd.DatetimeIndex(Tesla["Date"])
Tesla.set_index("Date", inplace=True)

Tesla["log_volume"] = np.log(Tesla["Volume"])
Tesla["log_return"] = Tesla["log_volume"] - Tesla["log_volume"].shift(1)
Tesla["log_return"].plot()
```

b) Berechnen Sie die Autokorrelationsfunktion für ein verrauschtes linear steigendes Signal.

```
from scipy import signal
import matplotlib.pyplot as plt
import numpy as np
t = np.linspace(0, 1, 1000, endpoint=False)
noise = np.random.normal(size=1000)
signal = 0.5*t
plt.plot(t, signal + noise)
plt.ylim(-2, 2)
```

c) Berechnen Sie die Autokorrelationsfunktion für ein verrauschtes Cosinus-Signal.

```
from scipy import signal
import matplotlib.pyplot as plt
import numpy as np
t = np.linspace(0, 1, 1000, endpoint=False)
noise = np.random.normal(size=1000)
signal = np.cos(2 * np.pi * 20 * t)
plt.plot(t, signal + noise)
plt.ylim(-2, 2)
```

d) Berechnen Sie ein verrauschtes Rechtecksignal.

```
from scipy import signal
import matplotlib.pyplot as plt
import numpy as np
t = np.linspace(0, 1, 1000, endpoint=False)
noise = np.random.normal(size=1000)
plt.plot(t, signal.square(2 * np.pi * 20 * t) + noise)
plt.ylim(-2, 2)
```

e) Berechnen Sie ein verrauschtes Sägezahnsignal.

```
from scipy import signal
import matplotlib.pyplot as plt
import numpy as np
t = np.linspace(0, 1, 1000, endpoint=False)
noise = np.random.normal(size=1000)
signal = signal.sawtooth(2 * np.pi * 20 * t)
plt.plot(t, signal + noise)
plt.ylim(-2, 2)
```

Aufgabe 13.3

a) Ein stochastischer Prozess sei definiert durch

$$X_i = T + (1 - i)$$

wobei T eine über dem Intervall $[0, 1]$ uniform verteilte Zufallsvariable ist. Bestimmen Sie die Autokorrelationsfunktion $\gamma(i, j) = \mathrm{E}[(X_i - \mu(i))(X_j - \mu(j))]$. Handelt es sich bei X_i um einen stationären stochastischen Prozess?

b) Wir betrachten den zeit-diskreten Zufallsprozess

$$X_n = A^n \,,$$

wobei A eine auf dem Intervall $[0,1]$gleichmässig verteilte Zufallsvariable ist und $n \in \mathbb{N}_0^+$. Handelt es sich bei X_n um einen stationären Zufallsprozess? Berechnen Sie dazu $\mu(n) = \mathrm{E}[X_n]$ und $\gamma(n,m) = E[(X_n - \mu(n))(X_m - \mu(m))]$.

Aufgabe 13.4

Wir betrachten den folgenden moving average Prozess

$$X_i = W_{i-1} + 2W_i + W_{i+1},$$

wobei W_i unabhängige Zufallsvariablen mit Erwartungswert 0 und Varianz σ^2 seien.

a) Berechnen Sie die Mittelwertsfolge dieses Prozesses.

b) Berechnen Sie die Autokovarianz und Autkorrelationsfolge dieses Prozesses.

c) Zeichnen Sie die Autokorrelationsfunktion $\rho(i,j)$ als eine Funktion vom lag $h = i - j$ auf.

Aufgabe 13.5

In dieser Aufgabe betrachten wir einen simulierten diskreten Prozess $\{X_1, X_2, \ldots\}$, der durch folgende Konstruktionsregel gegeben ist

- Setzen Sie $X_1 = -1$
- Für jedes $k \geq 1$ werde eine faire Münze geworfen, wobei man für Kopf $D_k = 1$ und für Zahl $D_k = -1$ setzt. Definieren Sie

$$X_k = a + D_k + bD_{k-1}$$

für bestimmte Zahlen $a, b \in \mathbb{R}$.

a) Generieren Sie eine Zeitreihe $\{x_1, x_2, \ldots, x_{200}\}$ basierend auf diesem Prozess, und stellen Sie die dadurch generierte Zeitreihe mit den Parameterwerten $a = 2$ und $b = -7$ graphisch dar.

Hinweis: Der Münzwurf kann durch eine binomialverteilte Zufallsvariable mit $n = 1$ and $p = 0.5$ modelliert werden (also einer Bernoulli-verteilten Zufallsvariablen).

Python erlaubt das Generieren von binomialverteilten Zufallsvariablen mit Hilfe des Befehls `np.random.binomial(size=..., n=..., p= ...)`.

Mit dem folgenden Code kann der Prozess in **Python** implementiert werden.

```
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.tsa.stattools import acf

# Create the process D_k
D = 2*(np.random.binomial(size=200, n=1, p= 0.5)-0.5)
# Create the process X_k
X=np.zeros(200)
X[0]=-1

for k in range(1,200):
    X[k] = 2 + D[k] - 0.7*D[k-1]
```

b) Berechnen Sie aufgrund der generierten Zeitreihe die empirische Autokorrelationsfunktion $\hat{\rho}(k)$ `acf()`. Erstellen Sie ein Korrelogramm bis zu lag 50.

c) Berechnen Sie den theoretischen Mittelwert $\mu(k)$ und die Autokorrelationsfunktion $\rho(k)$ des durch obige Regel definierten Prozesses (d.h. für allgemeine a und b). Vergleichen Sie $\hat{\rho}(1)$ und $\rho(1)$ für $a = 2$ und $b = -0.7$.

d) Ist der Prozess X_k schwach stationär?

Aufgabe 13.6

Wir werden uns in dieser Aufgabe mit einem wichtigen Datensatz auseinandersetzen : mit den monatliche Lufttemperaturmessungen auf der Erdoberfläche in der nördlichen Hemisphäre [1].

a) Laden Sie die Datei `global_temp.csv`

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
global_temp = pd.read_csv(".../global_temp.csv")
global_temp["Zeit"] = pd.DatetimeIndex(global_temp["Zeit"])
global_temp.set_index("Zeit", inplace=True)
print(global_temp.head())
```

[1] Der Datensatz wird regelmässig aktualisiert und kann frei von der Webseite http://www.cru.uea.ac.uk/data/ heruntergeladen werden.

Stellen Sie die Zeitreihe graphisch dar.

b) Führen Sie Zerlegung der Zeitreihe mit Hilfe von `seasonal_decompose` durch. Stellen Sie den Parameter `freq` optimal ein, indem Sie unterschiedliche Werte graphisch prüfen. Was beobachten Sie?

c) Wird die Restreihe durch einen schwach stationären Prozess generiert? Berechnen Sie das entsprechende Korrelogramm. Gibt es statistisch signifikante Korrelationen?

Kurzlösungen vereinzelter Aufgaben

A 13.3:

a) $\gamma(i,j) = \frac{13}{12}$ und nichtstationär

b) $\gamma(n,m) = \frac{1}{n+m+1} - \frac{1}{(n+1)\cdot(m+1)}$ und nicht-stationär

A 13.4:

$$\gamma(i,j) = \begin{cases} 6\sigma^2 & \text{falls } j = i \\ 4\sigma^2 & \text{falls } j = i+1 \\ \sigma^2 & \text{falls } j = i+2 \\ 0 & \text{ansonsten.} \end{cases}$$