REPORT ON ESTIMATION OF THE ARMA MODEL

Sumit Chowdhury, Indian Institute of Foreign Trade, Masters in Economics , 2021-23

August 2, 2023

[Maximum time available: 2 Weeks]

1 Characteristics of GDP Per Capita in USA

1.1 Checking the null values

```
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   YEARS                           63 non-null     int64
 1   GDP per capita (constant 2015 US$)  63 non-null     float64
```

The complete dataset of the USA, extracted from the metadata, does not contain any missing values. Therefore, we can confidently proceed with this dataset for further analysis.

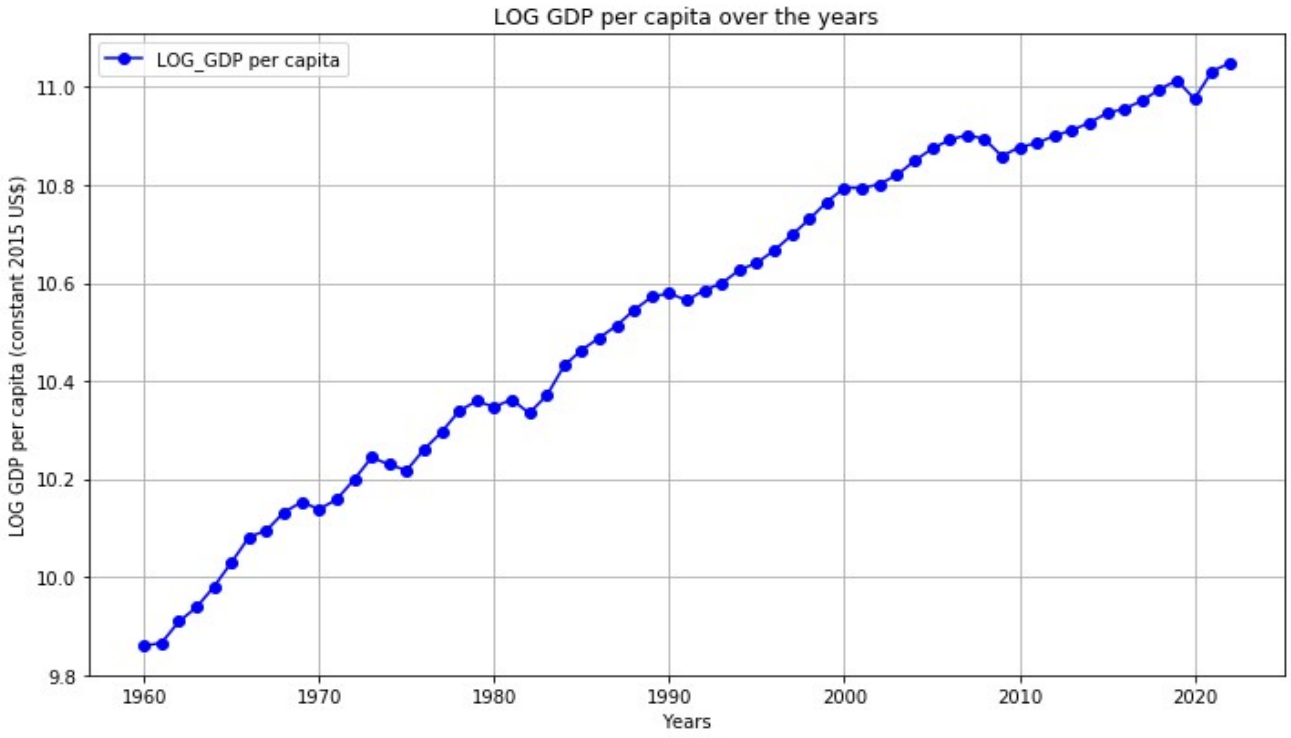
1.2 Taking the logarithm of GDP per capita

	YEARS	GDP per capita (constant 2015 US$)	log_GDP
0	1960	19135.26818	9.859288
1	1961	19253.54733	9.865451
2	1962	20116.23512	9.909282
3	1963	20701.26995	9.937950
4	1964	21599.81871	9.980440
...	...	...	...
58	2018	59607.39366	10.995535
59	2019	60698.01130	11.013666
60	2020	58451.60672	10.975954
61	2021	61829.84563	11.032141
62	2022	62866.71439	11.048772

1.3 Descriptive Statistics

	YEARS	GDP per capita (constant 2015 US$)	log_GDP
count	63.000000	63.000000	63.000000
mean	1991.000000	40172.301052	10.544309
std	18.330303	13070.714607	0.347295
min	1960.000000	19135.268180	9.859288
25%	1975.500000	28348.802190	10.252306
50%	1991.000000	39303.489010	10.579069
75%	2006.500000	52876.597795	10.875715
max	2022.000000	62866.714390	11.048772

1.4 Plotting log(GDP per capita) with respect to time



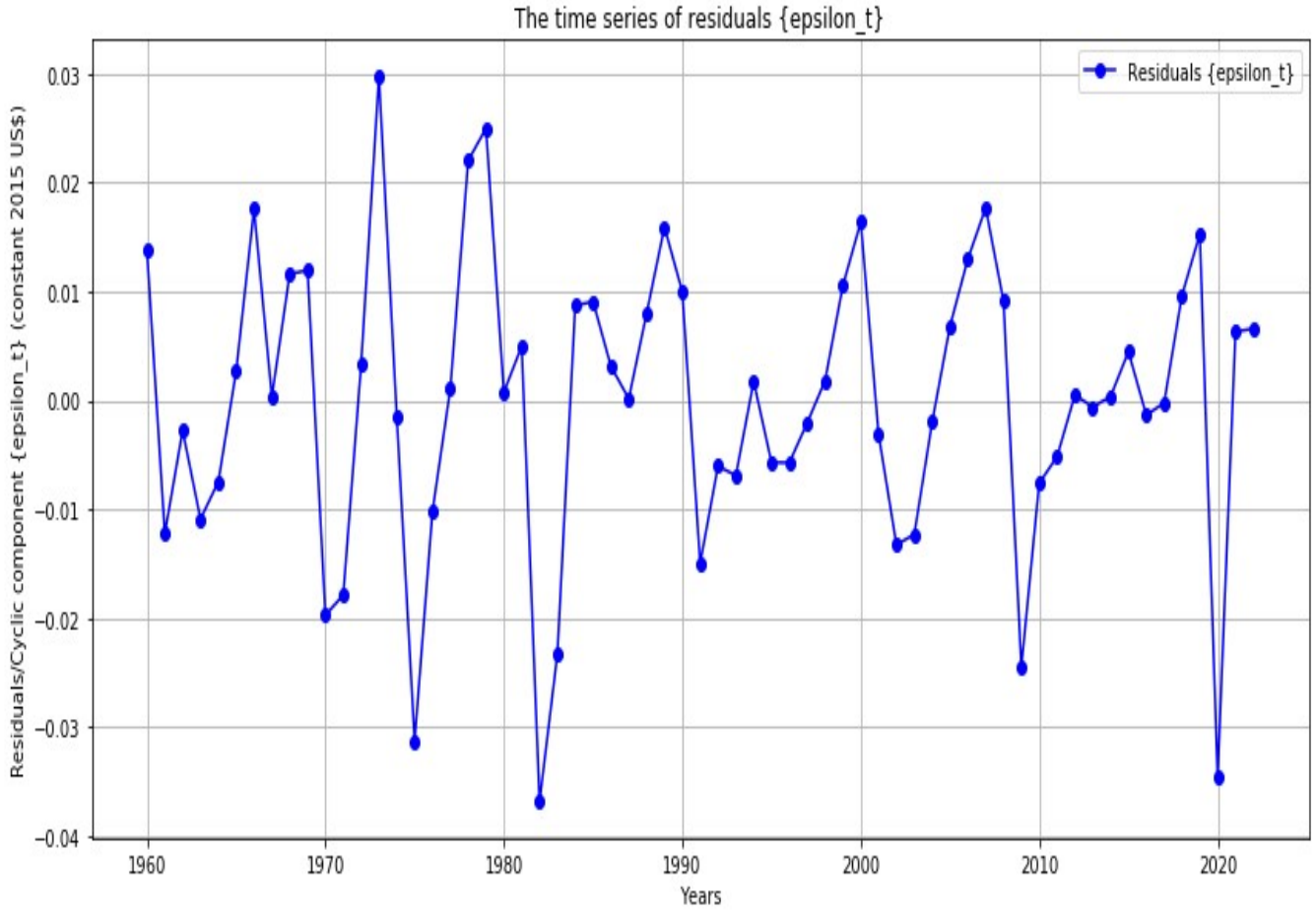
The plotted graph of log(GDP per capita) across the years clearly exhibits a prominent positive trend in the data. This strong indication of a trend suggests that the variable is not stationary. There is a high possibility that the mean or variance is influenced by time, and cyclic patterns may also be present in the data. To address these complexities, the use of HP-filtering becomes necessary to separate the time component from its cyclical counterpart.

1.5 Filtering the series using Hodrick-Prescott filter

	log_GDP	YEARS	epsilon_t	trend_t
0	9.859288	1960	0.013764	9.845525
1	9.865451	1961	-0.012229	9.877680
2	9.909282	1962	-0.002755	9.912037
3	9.937950	1963	-0.010892	9.948842
4	9.980440	1964	-0.007459	9.987899
...	...	...	...	...
58	10.995535	2018	0.009652	10.985883
59	11.013666	2019	0.015275	10.998391
60	10.975954	2020	-0.034596	11.010551
61	11.032141	2021	0.006309	11.025833
62	11.048772	2022	0.006601	11.042171

After employing the HP filter, we can distinctly observe the trend and cyclical aspects of log GDP per capita. We consider the cyclical component as the residual concerning the trend component. The smoothing parameter of 6.25 is utilized, considering the data's yearly nature. To facilitate analysis, we refer to the residual part as $epsilon_t$.

1.6 Plotting the time series of the residual component



Based on the time series plot of the residuals, there is no discernible trend, and the nature appears to be random. This suggests that the variable might possess the characteristics of a stationary time series, where the mean and variance remain constant over time. However, a closer inspection of the variance indicates that the amplitude is not consistent over time, raising doubts about the series' stationarity.Further analysis should be needed for model selection. To confirm whether the series is indeed stationary, it is advisable to conduct the Augmented Dickey-Fuller test on the data. This test will provide a definitive assessment of the stationarity of the series.

1.7 Checking the stationarity of residuals - ADF TEST

```
ADF Statistic: -4.194677858895401
p-value: 0.0006727025841279763
```

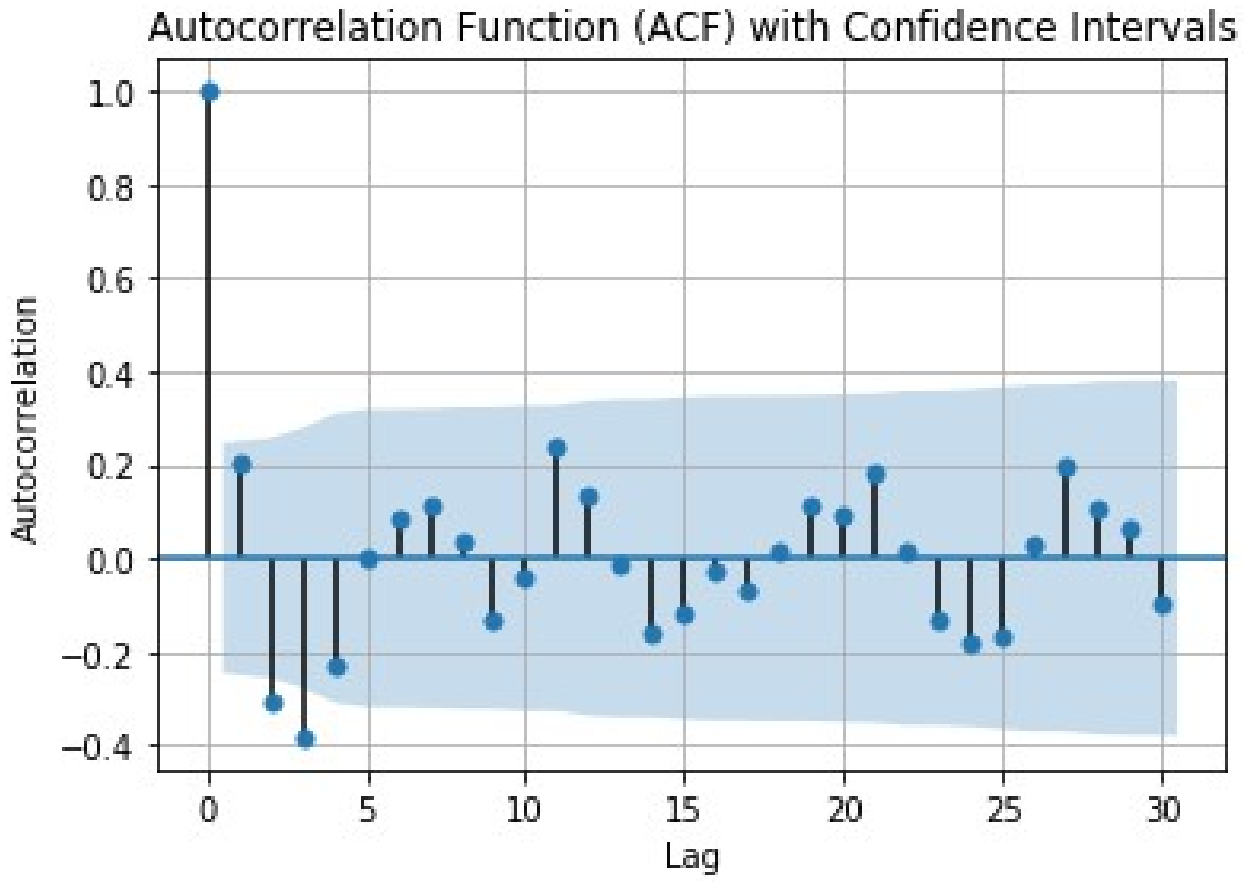
The ADF Statistic value is -4.194677858895401. It is more negative than the critical values

at all significance levels.The p-value is also very small (0.0006727025841279763).

Since the ADF Statistic is more negative than the critical values, and the p-value is significantly lower than the chosen significance level (e.g., 0.05), we can reject the null hypothesis.

Based on the ADF test results, we can conclude that the time series is stationary. The mean and variance of the series remain constant over time, and there is no unit root, indicating stationarity.

1.8 Plotting empirical autocorrelation function (ACF) of residuals

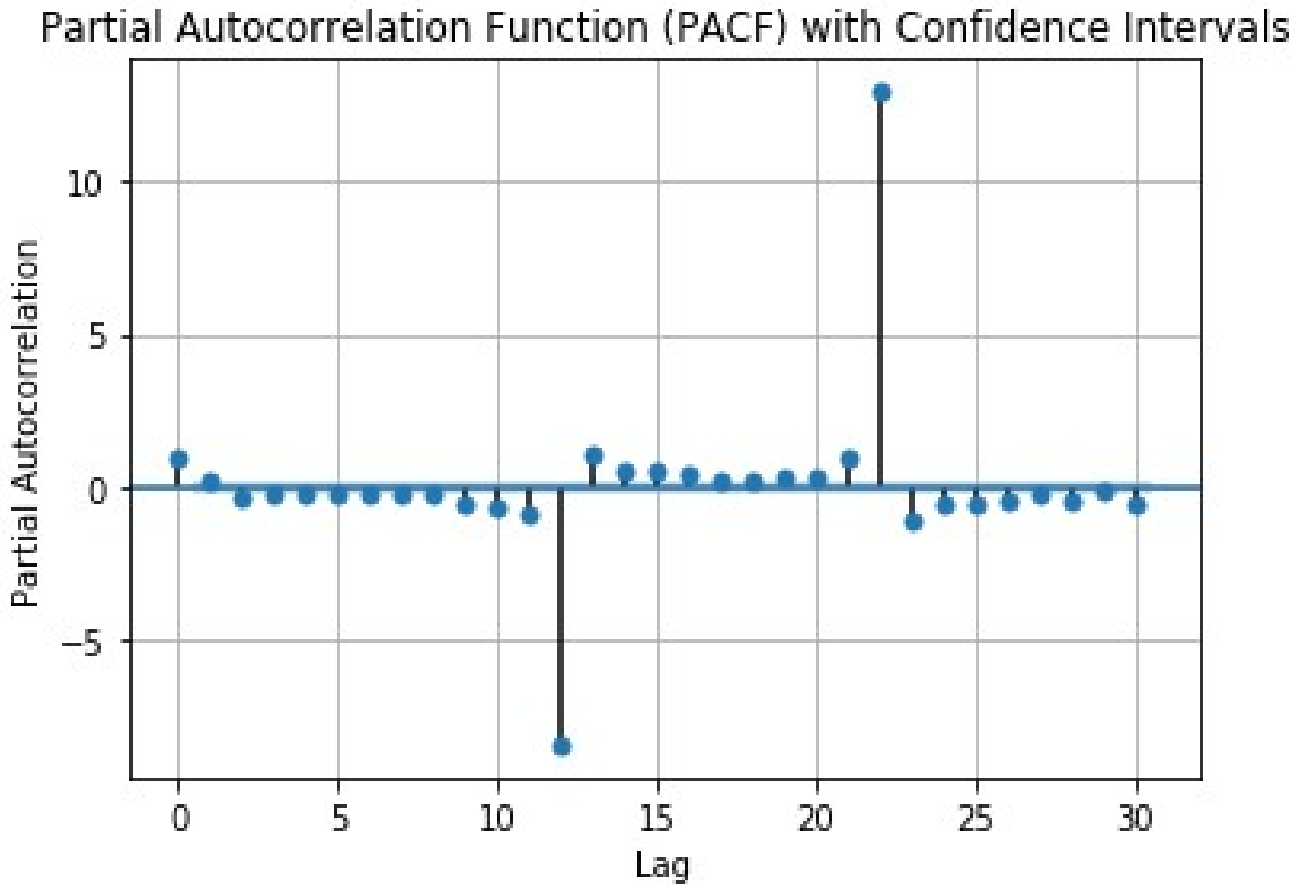


We first computed the ACF and PACF values for the $epsilon_t$ time series up to 30 lags. The ACF measures the correlation between the series and its lags, while the PACF measures the correlation between the series and its lags after removing the correlations explained by earlier lags.

The ACF plot displayed a significant pattern of correlation up to 3 lags and became

insignificant for all lags beyond 3. This is also clear from the graph that the ACF of $epsilon_t$ is geometrically declining, indicating autoregressive properties in the process. But the statement can be stictly validated only after looking at the plot of PACF.

1.9 Plotting empirical partial autocorrelation function (PACF) of residuals



The PACF plot showed notable spikes at lags 12 and 22. These spikes indicate strong partial autocorrelation at these specific lags, even after accounting for correlations explained by earlier lags.The significant PACF spikes at lags 12 and 22 suggest the presence of seasonality in the data. The significant ACF values up to lag 4 suggest the presence of short-term dependence or autoregressive effects in the data. The ACF dropping sharply to insignificance at lag 4 and beyond indicates that the autocorrelation dies out quickly after lag 3.

For a pure AR(p) process, the PACF exhibits a clear pattern of significant spikes at the first p lags (lags 1 to p) and then drops to zero or becomes insignificant for lags greater than

p. This is referred to as "PACF cuts off after lag p." But since this is not happening till lag 22 , hence we can say the process is not a pure AR process but it is likely that a MA component is also there in the time series .

1.10 Computing Akaike information criteria (AIC) matrix for choosing the correct ARMA Model

	0	1	2	3	4	5	6	7	8	9
0	-359.809105	-363.200853	-380.255379	-382.428368	-381.661308	NaN	NaN	-391.000495	-387.116198	-389.836474
1	-360.539828	-361.593892	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	-367.835276	-389.840413	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	-370.771766	-388.848020	-387.163229	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	-372.482005	-388.181872	-387.093091	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	-372.471438	-387.147266	-385.707813	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6	-373.489683	-387.017626	-385.512399	-385.772401	-384.095782	NaN	NaN	NaN	NaN	NaN
7	-372.294992	-385.947661	-384.456392	-385.332808	-383.073177	NaN	NaN	NaN	NaN	NaN
8	-373.105336	-387.157342	-384.289810	-381.291428	-381.744214	-383.969422	NaN	NaN	NaN	NaN
9	-379.308940	-386.981140	-385.179636	-385.299380	NaN	NaN	NaN	NaN	NaN	NaN

1.11 Model Selection

So from the AIC matrix we will choose the model which has the lowest value. The AIC is a useful tool for model selection as it balances the goodness of fit and model complexity. It helps to avoid overfitting and provides a way to compare different models to choose the one that provides the best trade-off between fit and complexity. Lower AIC values indicate better-performing models.

AIC penalizes models that use more parameters. So if two models explain the same

amount of variation, the one with fewer parameters will have a lower AIC score and will be the better-fit model.

Based on the AIC matrix, we can identify that the MA(7) model has the lowest AIC value. However, considering the clear presence of autoregressive properties in the data, this model is not suitable. Instead, we should search for the next best fit, which corresponds to the ARMA(2,1) model, as it has the next lowest AIC value in the matrix.

So one can choose *ARMA(2,1)* model as a preferred model based on AIC criterion.

2 Appendix (codes)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

df=pd.read_csv('C:/Users/Sumit/Downloads/USA_GDP_TIME_SERIES.csv')
## importing the dataset

df.info()
## importing the dataset

df['log_GDP']=np.log(df['GDP_per_capita_(constant_2015_US$)'])
## taking the log of GDP PER CAPITA
# Create the plot of log(GDP)
plt.figure(figsize=(10, 6))  # Adjust the figure size as needed

# Plot the data points as a line plot
plt.plot(df.YEARS, df.log_GDP, marker='o', linestyle='-', color='b'
,
label='LOG_GDP_per_capita')

# Add labels and title
```

```
plt.xlabel('Years')
plt.ylabel('LOG GDP per capita (constant 2015 US$)')
plt.title('LOG GDP per capita over the years')
plt.grid(True)  # Add grid lines for better readability
plt.legend()  # Show the legend

# Show the plot
plt.tight_layout()  # Ensure all elements are visible
plt.show()

## Filter the series using Hodrick-Prescott filter (HP filter)
from statsmodels.tsa.filters.hp_filter import hpfilter
log_gdp_cycle,log_gdp_trend = hpfilter(df['log_GDP'], lamb=6.25) #Use 6.25 as t

gdp_segr = df[['log_GDP']]
gdp_segr['YEARS'] = df['YEARS']
gdp_segr['epsilon_t']= log_gdp_cycle
gdp_segr['trend_t'] = log_gdp_trend
gdp_segr

# Create the plot of time series of residuals
plt.figure(figsize=(12, 6))  # Adjust the figure size as needed

# Plot the data points as a line plot
plt.plot(gdp_segr.YEARS, gdp_segr.epsilon_t, marker='o', linestyle='-', color='b

# Add labels and title
plt.xlabel('Years')
plt.ylabel('Residuals/Cyclic component {epsilon_t} (constant 2015 US$)')
plt.title('The time series of residuals {epsilon_t}')
plt.grid(True)  # Add grid lines for better readability
```

```
plt.legend()  # Show the legend

# Show the plot
plt.tight_layout()  # Ensure all elements are visible
plt.show()

#checking for stationarity–ADF TEST

result = sm.tsa.adfuller(gdp_segr['epsilon_t'])

print("ADF Statistic:", result[0])
print("p–value:", result[1])
print("Critical Values:", result[4])

from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

# Plot ACF with confidence intervals
plt.figure(figsize=(24,12))
plot_acf(gdp_segr['epsilon_t'], lags=30, alpha=0.05)
plt.xlabel('Lag')
plt.ylabel('Autocorrelation')
plt.title('Autocorrelation Function (ACF) with Confidence Intervals')
plt.grid(True)
plt.show()

# Plot PACF with confidence intervals
plt.figure(figsize=(24, 12))
plot_pacf(gdp_segr['epsilon_t'], lags=30, alpha=0.05)
plt.xlabel('Lag')
plt.ylabel('Partial Autocorrelation')
plt.title('Partial Autocorrelation Function (PACF) with Confidence Intervals')
```

```
plt.grid(True)
plt.show()

# Initialize AIC matrix
p_values = range(10)
q_values = range(10)
aic_matrix = [[0 for _ in range(10)] for _ in range(10)]

# Iterate through different combinations of p and q
for p in p_values:
    for q in q_values:
        try:
            # Estimate ARMA(p,q) model
            arma_model = sm.tsa.ARMA(gdp_segr['epsilon_t'], order=(p, q))
            arma_result = arma_model.fit()
            # Compute AIC
            aic_value = arma_result.aic
            # Store AIC in the matrix
            aic_matrix[p][q] = aic_value
        except:
            # In case of convergence issues, store AIC as None
            aic_matrix[p][q] = None

# Converting the matrix to a DataFrame for better visualization
aic_df = pd.DataFrame(aic_matrix, index=p_values, columns=q_values)
print(aic_df)
```