DATS 6313 – Time Series Analysis & Modeling

Instructor: Reza Jafari

Lab #7

Bradley Reardon

3/29/2022

Abstract:

The lab is about Generalized Partial Autocorrelation table for Autoregressive (AR) & Moving Average (MA) Model.

Introduction:

This experiment was performed to increase understanding of the application of GPAC and how the table can be converted into a python program.

Method, Theory, and Procedures:

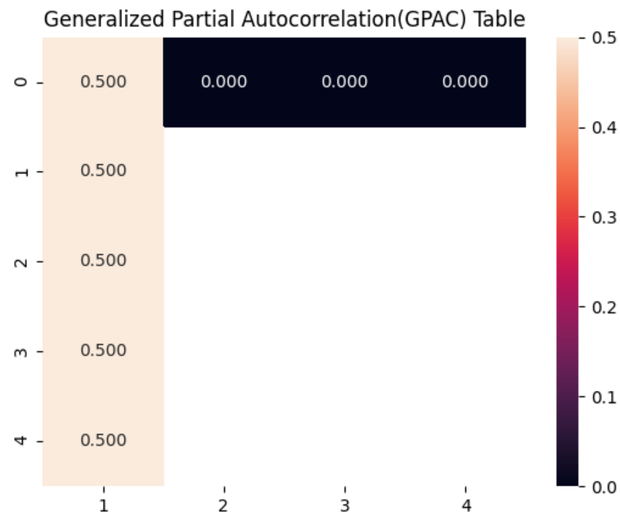
GPAC Table:

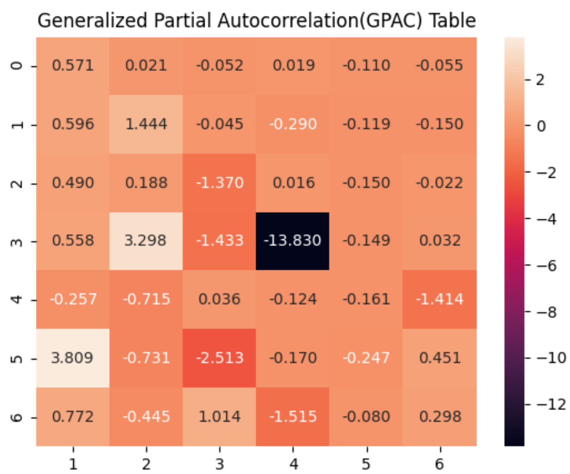
$$\phi_{kk}^{j} = \frac{\begin{vmatrix} \hat{R}_y(j) & \hat{R}_y(j-1) & \dots & \hat{R}_y(j+1) \\ \hat{R}_y(j+1) & \hat{R}_y(j) & \dots & \hat{R}_y(j+2) \\ \vdots & \vdots & \vdots & \vdots \\ \hat{R}_y(j+k-1) & \hat{R}_y(j+k-2) & \dots & \hat{R}_y(j+k) \end{vmatrix}}{\begin{vmatrix} \hat{R}_y(j) & \hat{R}_y(j-1) & \dots & \hat{R}_y(j-k+1) \\ \hat{R}_y(j+1) & \hat{R}_y(j) & \dots & \hat{R}_y(j-k+2) \\ \vdots & \vdots & \vdots & \vdots \\ \hat{R}_y(j+k-1) & \hat{R}_y(j+k-2) & \dots & \hat{R}_y(j) \end{vmatrix}}$$

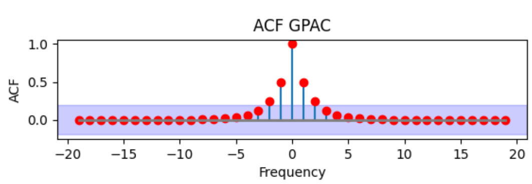
j \ k	1	2	$\cdots$	n_a			
0	ϕ_{11}^{0}	ϕ_{22}^{0}	$\cdots$				
1	ϕ_{11}^{1}	ϕ_{22}^{1}	$\cdots$				
$\vdots$	$\vdots$	$\vdots$					
n_b				$-a_{n_a}$	0	0	0
				$-a_{n_a}$	$\frac{0}{0}$	$\frac{0}{0}$	$\frac{0}{0}$
				$-a_{n_a}$	$\frac{0}{0}$	$\frac{0}{0}$	$\frac{0}{0}$

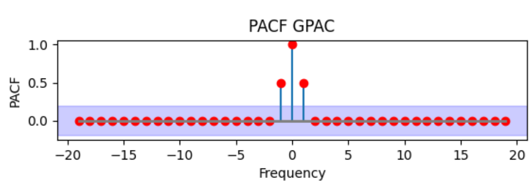
Answers to Lab Questions:

Example 1: $y(t) - 0.5y(t-1) = e(t)$ ARMA (1,0)

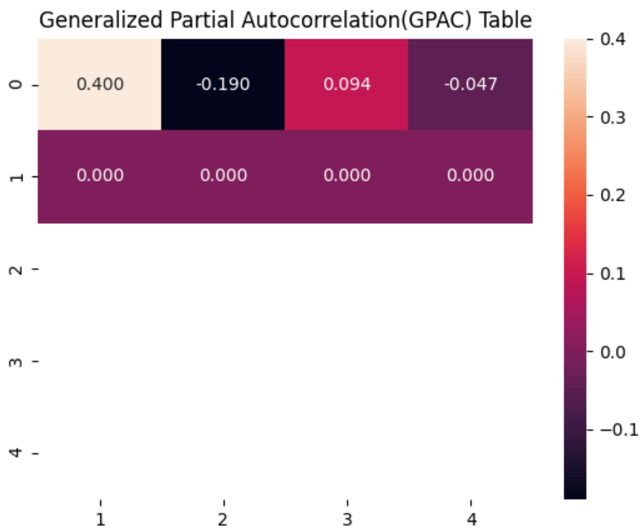


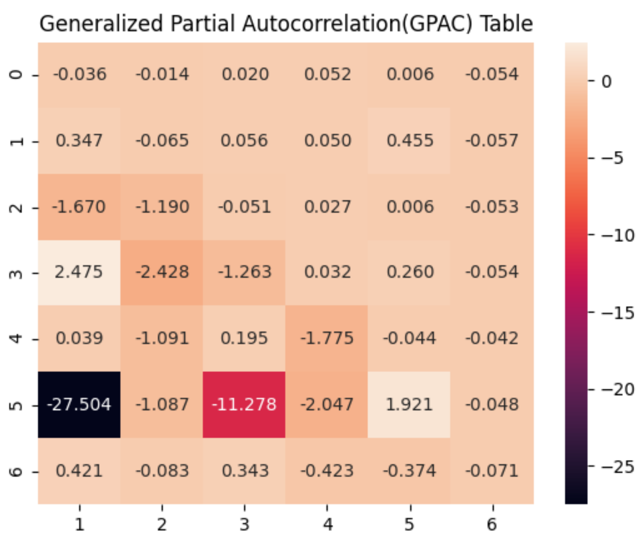


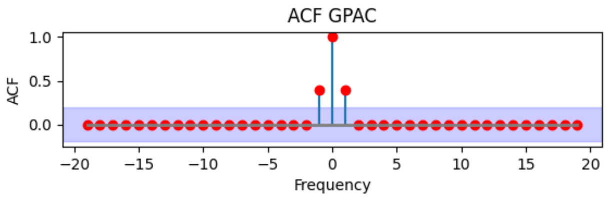


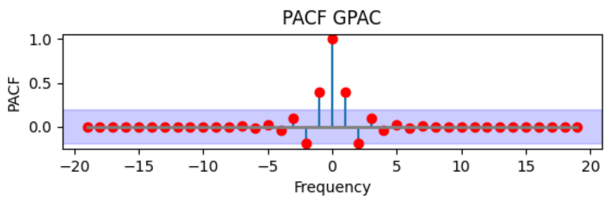


Example 2: ARMA (0,1): y(t) = e(t) + 0.5e(t-1)

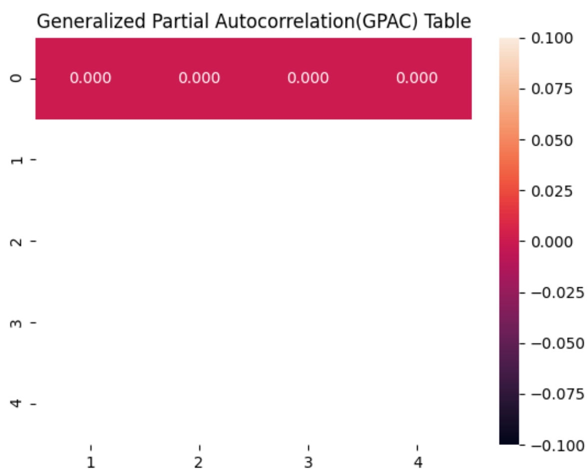

Generalized Partial Autocorrelation(GPAC) Table
0.400 -0.190 0.094 -0.047
0.000 0.000 0.000 0.000
0 1 2 3 4
1 2 3 4
0.4 0.3 0.2 0.1 0.0 −0.1

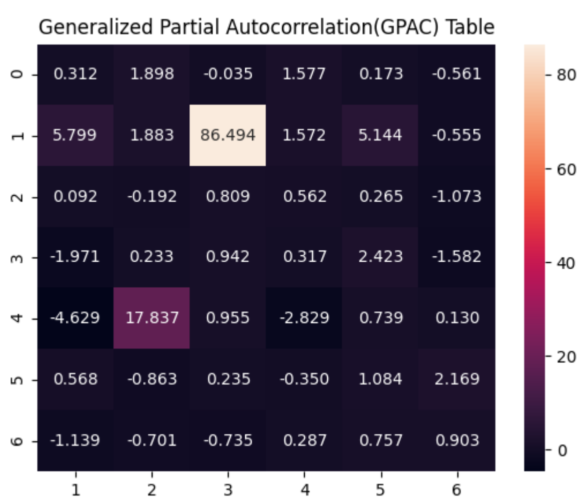

Generalized Partial Autocorrelation(GPAC) Table
-0.036 -0.014 0.020 0.052 0.006 -0.054
0.347 -0.065 0.056 0.050 0.455 -0.057
-1.670 -1.190 -0.051 0.027 0.006 -0.053
2.475 -2.428 -1.263 0.032 0.260 -0.054
0.039 -1.091 0.195 -1.775 -0.044 -0.042
-27.504 -1.087 -11.278 -2.047 1.921 -0.048
0.421 -0.083 0.343 -0.423 -0.374 -0.071
0 1 2 3 4 5 6
1 2 3 4 5 6
0 −5 −10 −15 −20 −25

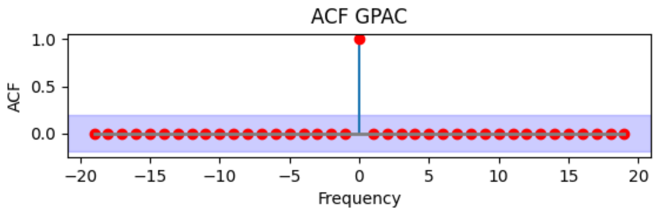


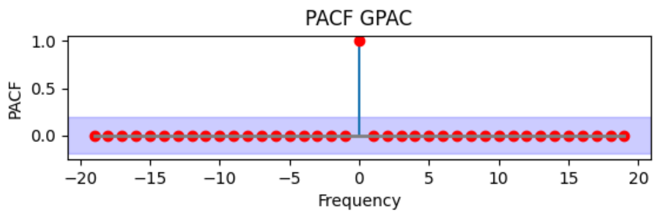


Example 3: ARMA (1,1): y(t) + 0.5y(t-1) = e(t) + 0.5e(t-1)

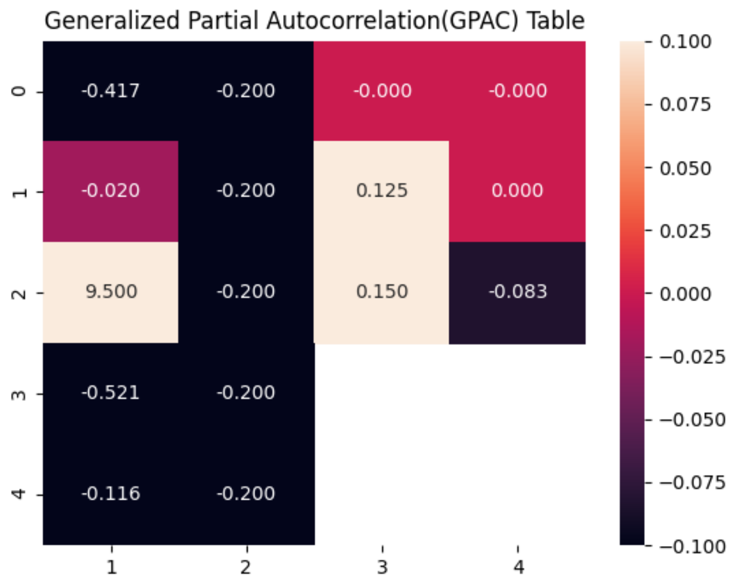


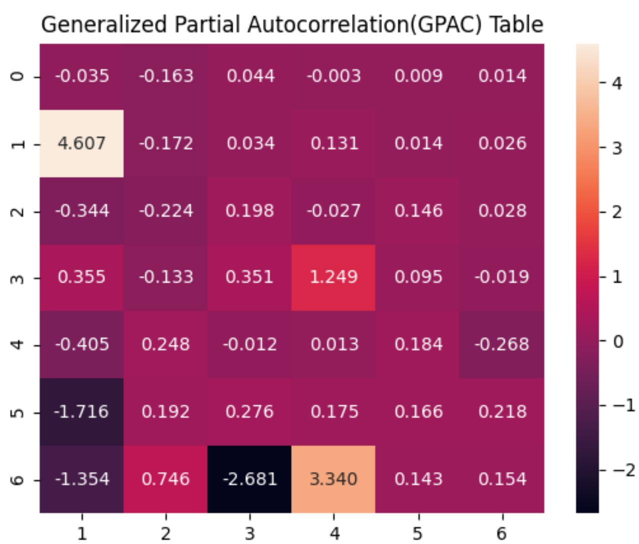


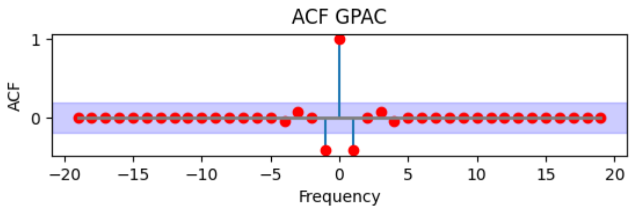


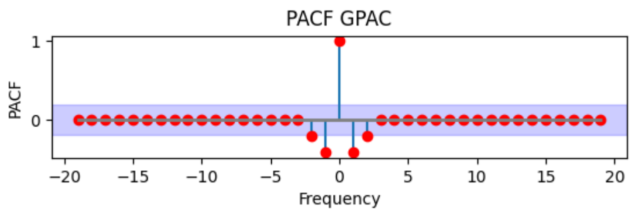


Example 4: ARMA (2,0): y(t) + 0.5y(t-1) + 0.2y(t-2) = e(t)

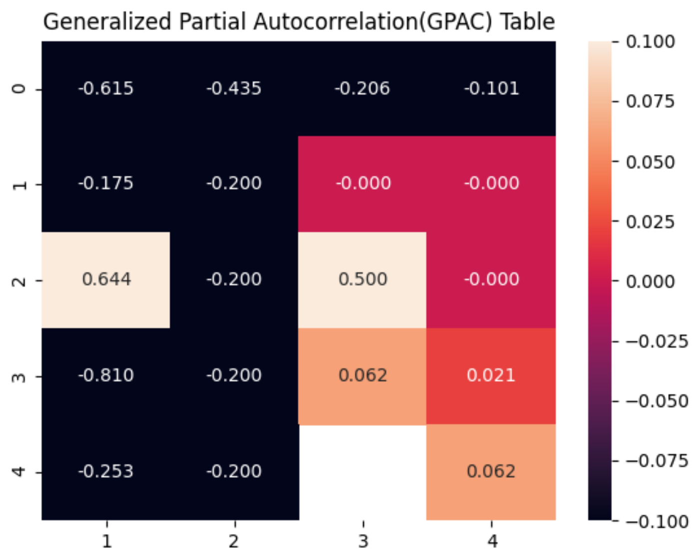

Generalized Partial Autocorrelation(GPAC) Table
-0.417
-0.200
-0.000
-0.000
-0.020
-0.200
0.125
0.000
9.500
-0.200
0.150
-0.083
-0.521
-0.200
-0.116
-0.200
0
1
2
3
4
1
2
3
4
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.100

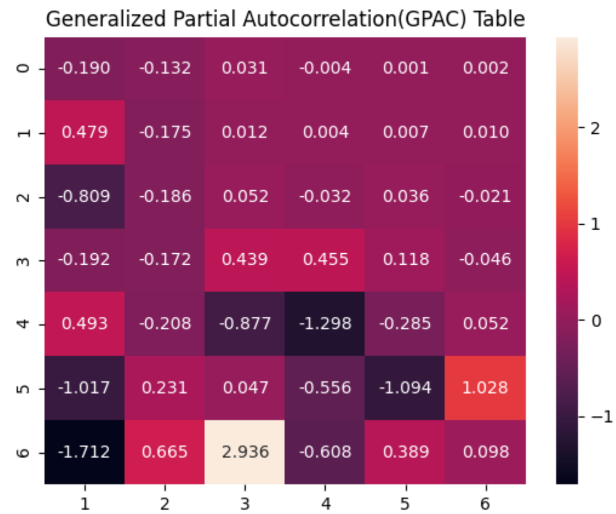

Generalized Partial Autocorrelation(GPAC) Table
-0.035
-0.163
0.044
-0.003
0.009
0.014
4.607
-0.172
0.034
0.131
0.014
0.026
-0.344
-0.224
0.198
-0.027
0.146
0.028
0.355
-0.133
0.351
1.249
0.095
-0.019
-0.405
0.248
-0.012
0.013
0.184
-0.268
-1.716
0.192
0.276
0.175
0.166
0.218
-1.354
0.746
-2.681
3.340
0.143
0.154
0
1
2
3
4
5
6
1
2
3
4
5
6
4
3
2
1
0
−1
−2

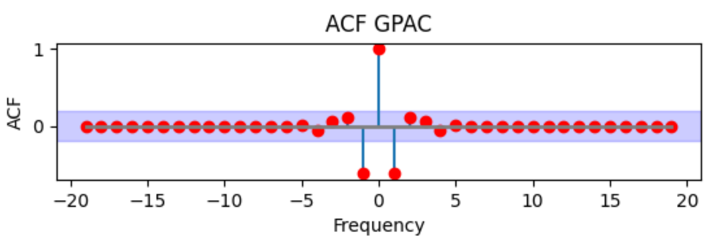


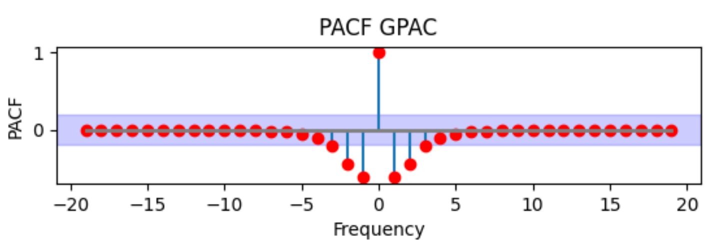


Example 5: ARMA (2,1): y(t) + 0.5y(t-1) + 0.2y(t-2) = e(t) - 0.5e(t-1)

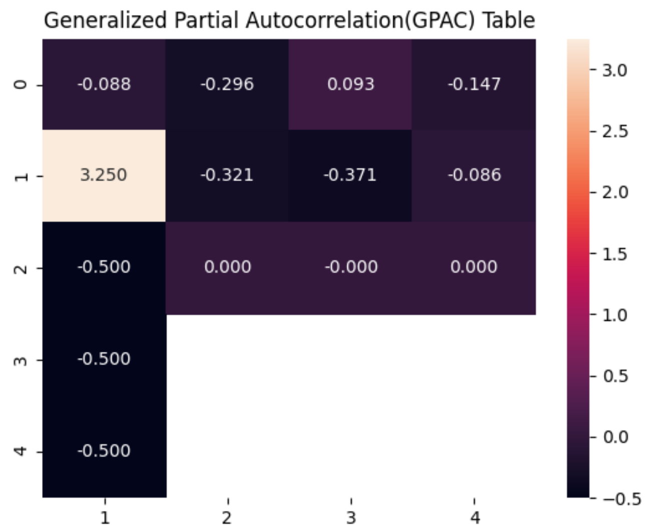


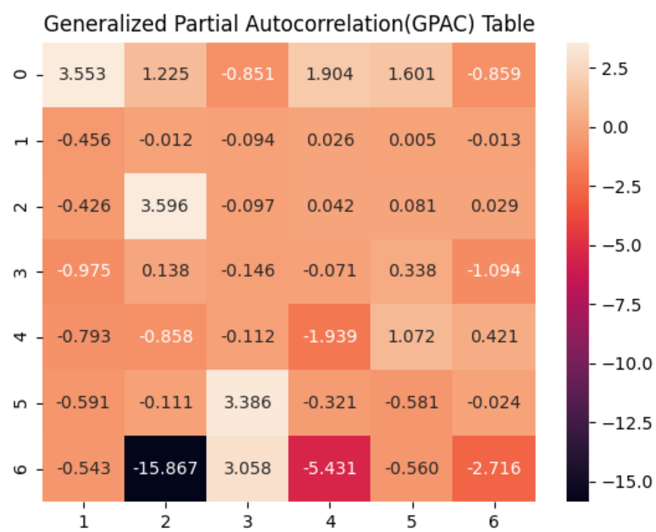


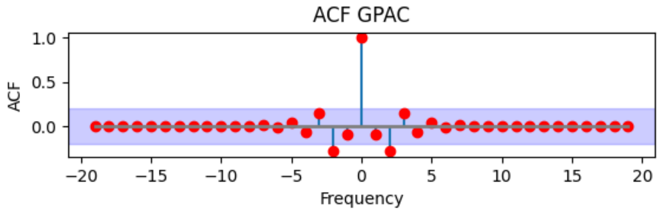


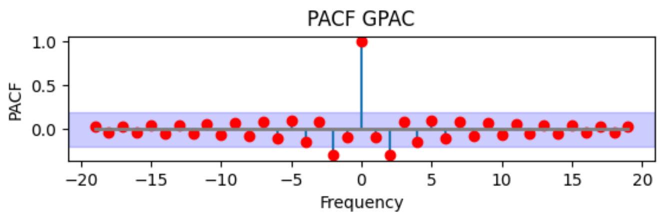


Example 6: ARMA (1,2): y(t) + 0.5y(t-1) = e(t) + 0.5e(t-1) - 0.4e(t-2)

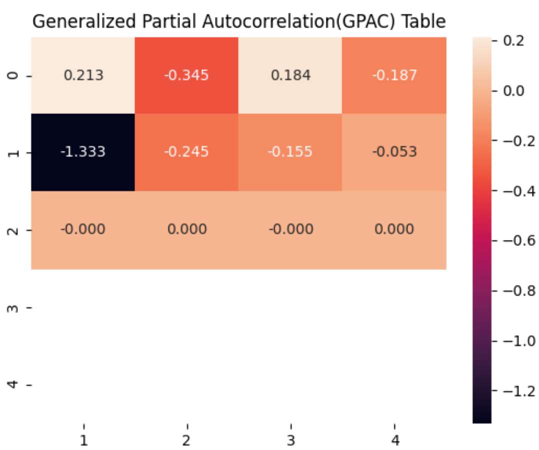

Generalized Partial Autocorrelation(GPAC) Table
-0.088 -0.296 0.093 -0.147
3.250 -0.321 -0.371 -0.086
-0.500 0.000 -0.000 0.000
-0.500
-0.500
0 1 2 3 4
1 2 3 4
3.0 2.5 2.0 1.5 1.0 0.5 0.0 −0.5

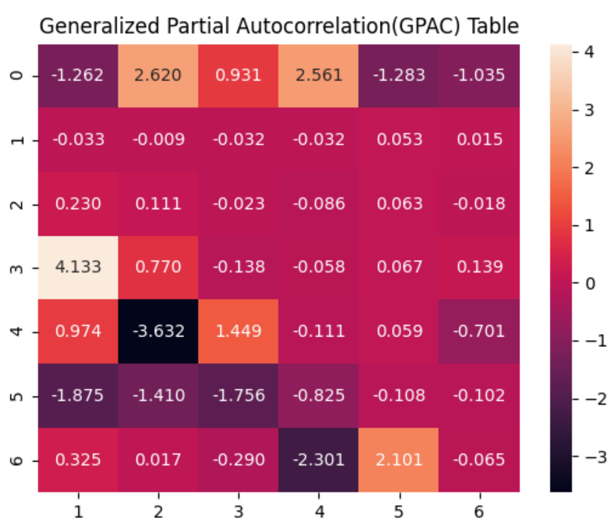

Generalized Partial Autocorrelation(GPAC) Table
3.553 1.225 -0.851 1.904 1.601 -0.859
-0.456 -0.012 -0.094 0.026 0.005 -0.013
-0.426 3.596 -0.097 0.042 0.081 0.029
-0.975 0.138 -0.146 -0.071 0.338 -1.094
-0.793 -0.858 -0.112 -1.939 1.072 0.421
-0.591 -0.111 3.386 -0.321 -0.581 -0.024
-0.543 -15.867 3.058 -5.431 -0.560 -2.716
0 1 2 3 4 5 6
1 2 3 4 5 6
2.5 0.0 −2.5 −5.0 −7.5 −10.0 −12.5 −15.0

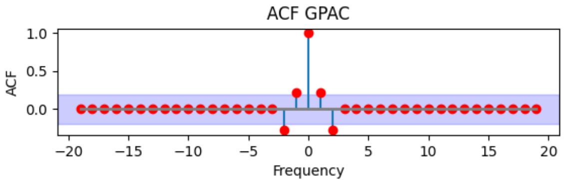


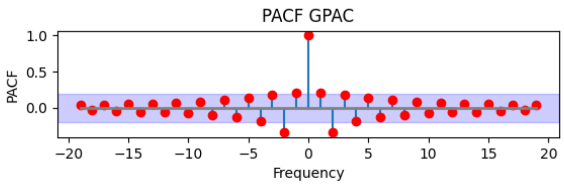


Example 7: ARMA (0,2): y(t) = e(t) + 0.5e(t-1) - 0.4e(t-2)

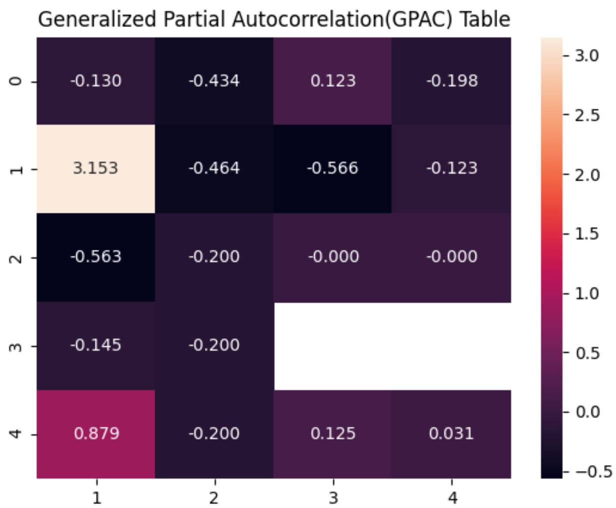


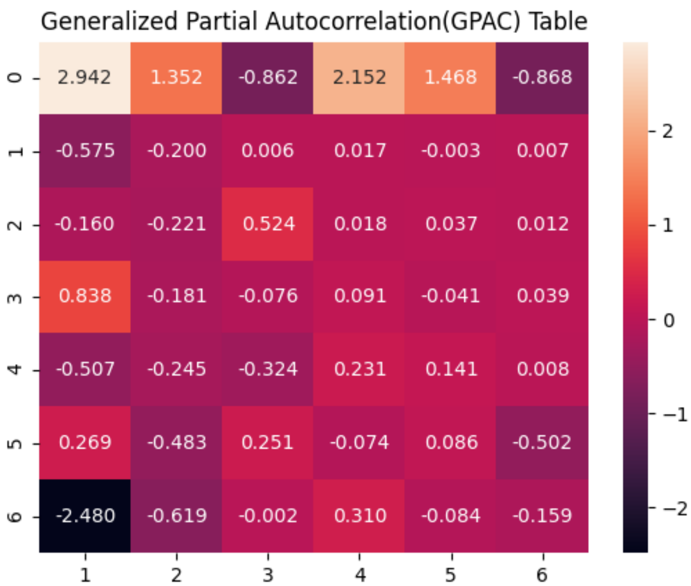


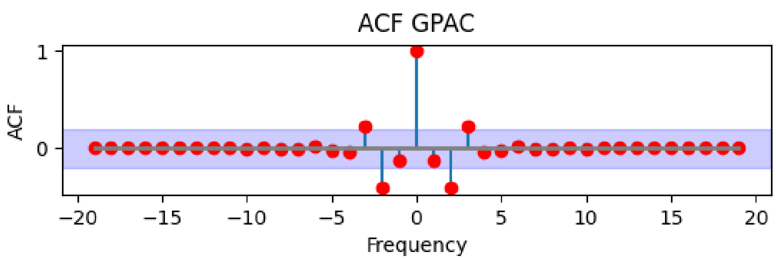


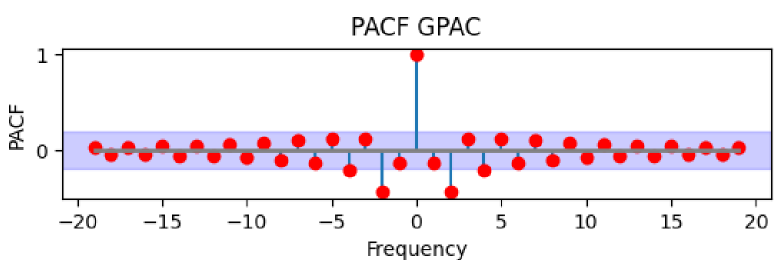


Example 8: ARMA (2,2): y(t)+0.5y(t-1) +0.2y(t-2) = e(t)+0.5e(t-1) - 0.4e(t-2)









8. It seems that higher order ARMA models have less static ACF and PACF graphs.

Conclusion:

GPAC charts can be used to determine the order of an ARMA model.

Appendix:

```python
import numpy as np
import statsmodels.api as sm
from matplotlib import pyplot as plt
import toolbox

print("====================QUESTION 1==========================")
samples_size = input("Enter # of samples: ")
mean = input("Enter Mean (WN): ")
var = input("Enter Variance (WN): ")
ar_order = input("Enter AR order: ")
ma_order = input("Enter MA order: ")

ar_inputs = []
if int(ar_order) > 0:
    for i in range(int(ar_order)):
        text = "Enter a" + str(i + 1) + ": "
        input_value = input(text)
        ar_inputs.append(float(input_value))

ma_inputs = []
if int(ma_order) > 0:
    for i in range(int(ma_order)):
        text = "Enter b" + str(i + 1) + ": "
        input_value = input(text)
        ma_inputs.append(float(input_value))

ar = np.r_[1, ar_inputs]
ma = np.r_[1, ma_inputs]

samples_size = int(samples_size)
ar_order = int(ar_order)
ma_order = int(ma_order)
mean = float(mean)
var = float(var)

arma_process = sm.tsa.ArmaProcess(ar, ma)
mean_y = mean * (1 + np.sum(ma_inputs)) / (1 + np.sum(ar_inputs))
y = arma_process.generate_sample(samples_size, scale=np.sqrt(var) + mean_y)


acf_lags = 60
ry = arma_process.acf(lags=acf_lags)
toolbox.gpac_calc(ry, 5, 5)

acf_lags = 15
new_ry = toolbox.auto_correlation_cal(y, acf_lags)
toolbox.gpac_calc(new_ry, 7, 7)

lags = 20

acf = arma_process.acf(lags=lags)
pacf = arma_process.pacf(lags=lags)

fig, axs = plt.subplots(2, 1)
fig.subplots_adjust(hspace=1.5, wspace=0.5)
axs = axs.ravel()
```

```
ry = arma_process.acf(lags=lags)
a1 = ry
a2 = a1[::-1]
a = np.concatenate((a2[:-1], a1))
x1 = np.arange(0, lags)
x2 = -x1[::-1]
x = np.concatenate((x2[:-1], x1))
(marker, stemlines, baselines) = axs[0].stem(x, a,
                                             use_line_collection=True,
markerfmt='o')
plt.setp(marker, color='red', marker='o')
plt.setp(baselines, color='gray', linewidth=2, linestyle='-')
m = 1.96 / np.sqrt(100)
axs[0].axhspan(-m, m, alpha=.2, color='blue')
axs[0].set_title("ACF GPAC")
axs[0].set_ylabel("ACF")
axs[0].set_xlabel("Frequency")

ry = arma_process.pacf(lags=lags)
a1 = ry
a2 = a1[::-1]
a = np.concatenate((a2[:-1], a1))
x1 = np.arange(0, lags)
x2 = -x1[::-1]
x = np.concatenate((x2[:-1], x1))
(marker, stemlines, baselines) = axs[1].stem(x, a,
                                             use_line_collection=True,
markerfmt='o')
plt.setp(marker, color='red', marker='o')
plt.setp(baselines, color='gray', linewidth=2, linestyle='-')
m = 1.96 / np.sqrt(100)
axs[1].axhspan(-m, m, alpha=.2, color='blue')
axs[1].set_title("PACF GPAC")
axs[1].set_ylabel("PACF")
axs[1].set_xlabel("Frequency")

plt.show()


samples_size = 5000
arma_process = sm.tsa.ArmaProcess(ar, ma)
mean_y = mean * (1 + np.sum(ma_inputs)) / (1 + np.sum(ar_inputs))
arma_process.generate_sample(samples_size, scale=np.sqrt(var) + mean_y)


acf_lags = 60
ry = arma_process.acf(lags=acf_lags)
toolbox.gpac_calc(ry, 5, 5)


samples_size = 10000
arma_process = sm.tsa.ArmaProcess(ar, ma)
mean_y = mean * (1 + np.sum(ma_inputs)) / (1 + np.sum(ar_inputs))
arma_process.generate_sample(samples_size, scale=np.sqrt(var) + mean_y)

acf_lags = 60
```

```
ry = arma_process.acf(lags=acf_lags)
toolbox.gpac_calc(ry, 5, 5)
```