TIME SERIES MODELING & ANALYSIS

Instructor Name: Reza Jafari

Lab#: 9

Submitted by Dinesh Kumar Padmanabhan

Date: 20-Nov-2020

ABSTRACT

The main purpose of this LAB is to implement the GPAC array covered in lecture using Python program and test the accuracy of your code using an ARMA(na,nb) model. Using statsmodels we simulated the ARMA process. GPAC code that generates GPAC table for various numbers of rows and columns was coded from scratch and was shown that the General Partial Autocorrelation Function (GPAC), which has recently been suggested to be used as one of a set of convenient tools for order identification in ARMA models.

INTRODUCTION

The generalized partial autocorrelation is used to estimate the order of ARMA model when na != 0 and nb != 0. The generalized partial autocorrelation (GPAC) function was introduced by Woodward and Gray (1981) for purposes of model identification in the ARMA(p,q) setting. The GPAC function is an extension of the partial autocorrelation function used by Box and Jenkins (1975) in ARMA model identification. Woodward and Gray (1981) used an array to present the information in the GPXC function, and this array was shown to be related to the S-array of Gray, Kelley, and McIntire (1978). Woodward and Gray (1981) showed that the GPAC array uniquely determines p and q when the true autocorrelation is known, a property it shares with the S-array. Unique identification of p and q when the true autocorrelation function is known is only assured using the Box-Jenkins approach when either p=O or q=O. Woodward and Gray (1981) discussed the use of the GPAC based on single, finite length realizations, and showed examples in which the model identifying pattern in the GPAC was clearly discernible. Davies and Petruccelli (1984) presented simulation evidence and real data examples to argue that the sample GPAC array is unstable when applied to time series of only moderate length and that its use in detecting MA components is limited.

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

$j \backslash k$	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
			$-a_{n_a}$	$\frac{0}{0}$	$\frac{0}{0}$	$\frac{0}{0}$
			$-a_{n_a}$	$\frac{0}{0}$	$\frac{0}{0}$	$\frac{0}{0}$

METHOD, THEORY & PROCEDURES

Method:

1. Programming Language: Python

Libraries used: Some basic libraries used for analysis & model building are mentioned below

library(Numpy) - large collection of high-level mathematical functions to operate on these arrays.

library (Pandas) – For Data manipulation and analysis

library(Matplotlib) – is a system for declaratively creating graphics

library(Math) –To Compute mathematical calculations

library (statsmodels) – Import statistical models

library (scipy) – Scientific Computations

Theory:

To Implement GPAC array.

Procedure:

I shall be looking at the results of GPAC models and Perform various plots and infer about it in my analysis. And through my exploration I shall try to identify which methods perform better and draw inferences.

The Dataset will be explored in following stages:

1. **Data Exploration (EDA)** – looking at the models and making inferences about the data.
2. **Data Visualization** – Plotting different time series plots for the regression method and forecast accuracy.
3. **Testing** – Running Autocorrelation, Pearson correlation test to identify the correlation between errors.

ANSWERS TO QUESTIONS

File - unknown

```
C:\ProgramData\Anaconda3\python.exe "C:\Program Files\JetBrains\PyCharm 2019.3.1\plugins\python\helpers\pydev\pydevconsole.py" --mode=client --port=57324

import sys; print('Python %s on %s' % (sys.version, sys.platform))
sys.path.extend(['C:\\Users\\nsree_000\\Desktop\\Python-Quiz', 'C:/Users/nsree_000/Desktop/Python-Quiz'])

Python 3.7.4 (default, Aug  9 2019, 18:34:13) [MSC v.1915 64 bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 7.8.0 -- An enhanced Interactive Python. Type '?' for help.
PyDev console: using IPython 7.8.0

Python 3.7.4 (default, Aug  9 2019, 18:34:13) [MSC v.1915 64 bit (AMD64)] on win32
In[2]: runfile('C:/Users/nsree_000/Desktop/Python-Quiz/TIME SERIES/labs-911/ma.py', wdir='C:/Users/nsree_000/Desktop/Python-Quiz/TIME SERIES/labs-911')
--------------------□(□) − 0.5□(□ − 1) = □(□)-------------------
Enter the number of data samples :>? 1000
Enter AR order:>? 1
Enter MA order:>? 0
Enter Mean of white noise:>? 1
Enter variance of white noise:>? 2
Enter coefficient of AR a1>? -0.5
Is this stationary process:  True
GPAC Table for y:
         1        2        3        4        5        6        7
0  0.50600  0.05640  0.04877  0.02592 -0.00626 -0.00726 -0.01795
1  0.58893 -0.37989  0.01895  0.03781 -0.03704  0.00617 -0.03041
2  0.69463 -0.15370  0.49020  0.02500 -0.14286 -0.00000 -0.00000
3  0.73430  1.15102  0.24000  0.00000  0.00000      NaN      NaN
4  0.64474 -0.22695 -0.50000      NaN      NaN      NaN      NaN
5  0.60204 -1.60938 -1.00000      NaN      NaN      NaN      NaN
```

Page 1 of 7

```
6  0.42373  1.28155  2.33333      NaN      NaN      NaN      NaN
 Repeat for 5000 samples
Enter the number of data samples:>? 5000
GPAC Table for y:
          1        2        3        4        5        6        7
0   0.50000 -0.01467 -0.00656 -0.00441 -0.02780 -0.01250  0.01358
1   0.47800 -0.23818  0.00271  0.03763 -0.02617 -0.04392 -0.00415
2   0.45607 -0.33969  5.00000  0.00000 -0.04348 -0.07692  0.00000
3   0.42202 -2.61798 -2.00000      NaN -0.00000 -0.00000      NaN
4  -0.04348 -0.45494  0.30000      NaN      NaN      NaN      NaN
5  11.50000 -0.47170 -1.00000      NaN      NaN      NaN      NaN
6   0.52174  0.68000  0.33333      NaN      NaN      NaN      NaN
 Repeat for 10000 samples
Enter the number of data samples:>? 10000
GPAC Table for y:
         1        2        3        4        5        6        7
0  0.50800 -0.00682  0.00549  0.00157 -0.00878 -0.00249 -0.00912
1  0.49803  0.40119  0.00662  0.03125 -0.00752  0.03571 -0.01316
2  0.51383 -0.14778  0.50000 -0.00000 -0.00000 -0.00000 -0.00000
3  0.52308  2.83333  1.00000      NaN      NaN      NaN      NaN
4  0.42647 -0.18824 -1.00000      NaN      NaN      NaN      NaN
5  0.34483 -1.37500 -0.00000      NaN      NaN      NaN      NaN
6 -0.40000 -0.13636      NaN      NaN      NaN      NaN      NaN

--------------------y(t) = e(t) + 0.5e(t-1)--------------------
Enter the number of data samples :>? 10000
Enter AR order:>? 0
Enter MA order:>? 1
```

```
Enter Mean of white noise:>? 1
Enter variance of white noise:>? 2
coefficient of MA b1>? 0.5
Is this stationary process:  True
GPAC Table for y:
         1        2        3        4        5        6        7
0  0.39100 -0.19228  0.10767 -0.06502  0.01600 -0.02525  0.01194
1 -0.02558  0.01860 -0.00712 -0.03864 -0.08659 -0.01764 -0.00291
2 -0.80000  0.00660 -0.11321 -0.03571 -0.03226 -0.00000 -0.00000
3 -0.50000 -8.00000 -0.00000 -0.00000 -0.00000      NaN      NaN
4  4.50000 -1.50000      NaN      NaN      NaN      NaN      NaN
5  1.11111 -1.12500      NaN      NaN      NaN      NaN      NaN
6  0.35000 -0.11111      NaN      NaN      NaN      NaN      NaN
--------------------ARMA (1,1): y(t) + 0.5y(t-1) = e(t) + 0.5e(t-1)--------------------
Enter the number of data samples :>? 10000
Enter AR order:>? 1
Enter MA order:>? 1
Enter Mean of white noise:>? 1
Enter variance of white noise:>? 2
Enter coefficient of AR a1>? 0.5
coefficient of MA b1>? 0.5
Is this stationary process:  True
GPAC Table for y:
         1        2        3        4        5        6        7
0  0.01300  0.01283  0.00567 -0.00531  0.00198 -0.00496  0.00113
1  1.00000  0.00701  0.01764 -0.00377 -0.01010 -0.00404  0.00000
2  0.46154 -1.11111  0.00000 -0.00000 -0.00000 -0.00000      NaN
3 -0.83333 -0.10000      NaN      NaN      NaN      NaN      NaN
4 -0.40000  2.00000      NaN      NaN      NaN      NaN      NaN
5 -2.50000  1.00000      NaN      NaN      NaN      NaN      NaN
```

```
6 -0.20000 -0.00000      NaN      NaN      NaN      NaN      NaN
--------------------ARMA (2,0): y(t) + 0.5y(t-1) + 0.2y(t-2) = e(t)--------------------
Enter the number of data samples :>? 10000
Enter AR order:>? 2
Enter MA order:>? 0
Enter Mean of white noise:>? 1
Enter variance of white noise:>? 2
Enter coefficient of AR a1>? 0.5
Enter coefficient of AR a2>? 0.2
Is this stationary process:  True
GPAC Table for y:
          1        2        3        4        5        6        7
0  -0.41800 -0.22019 -0.01352 -0.00208 -0.01672 -0.00293 0.01455
1   0.01675 -0.19580  0.02055  0.10377 -0.01647 -0.08696 0.01117
2 -12.14286 -0.19562  2.55556  0.27273 -0.00000 -0.00000 -0.00000
3  -0.44706 -0.35345  0.02174 -0.00000      NaN      NaN      NaN
4   0.31579 -0.33740 -2.00000      NaN      NaN      NaN      NaN
5  -1.50000 -0.48193 -0.50000      NaN      NaN      NaN      NaN
6   0.33333 -1.12500 -0.00000      NaN      NaN      NaN      NaN
--------------------ARMA (2,1): y(t) + 0.5y(t-1) + 0.2y(t-2) = e(t) - 0.5e(t-1) --------------------
Enter the number of data samples :>? 10000
Enter AR order:>? 2
Enter MA order:>? 1
Enter Mean of white noise:>? 1
Enter variance of white noise:>? 2
Enter coefficient of AR a1>? 0.5
Enter coefficient of AR a2>? 0.2
coefficient of MA b1>? 0.5
Is this stationary process:  True
GPAC Table for y:
         1        2        3        4        5        6        7
0 -0.02800 -0.19994  0.09353 -0.06208  0.03039 -0.02608 -0.00457
1  7.10714 -0.21253 -0.03804 -0.01644 -0.02287 -0.03144 -0
```

```
.05382
2 -0.51256 -0.13260  0.09091  0.02151  0.01667 -0.04478  0.00000
3 -0.23529 -0.26465  0.19355 -0.00000  0.00000 -0.00000      NaN
4  0.37500 -0.08725  0.50000      NaN      NaN      NaN      NaN
5 -0.22222  1.15385  0.33333      NaN      NaN      NaN      NaN
6 -8.50000  1.73333  1.00000      NaN      NaN      NaN      NaN
--------------------ARMA (1,2): y(t) + 0.5y(t-1) = e(t) + 0.5e(t-1) - 0.4e(t-2) --------------------
Enter the number of data samples :>? 10000
Enter AR order:>? 1
Enter MA order:>? 2
Enter Mean of white noise:>? 1
Enter variance of white noise:>? 2
Enter coefficient of AR a1>? 0.5
coefficient of MA b1>? 0.5
coefficient of MA b2>? -0.4
Is this stationary process:  True
GPAC Table for y:
          1        2        3        4        5        6        7
0 -0.10200 -0.30053  0.10472 -0.15510  0.09856 -0.10584  0.06449
1  2.81373 -0.33289 -0.33551 -0.09015 -0.06639 -0.04652 -0.07007
2 -0.56794 -0.02778  0.04058  0.01353  0.03991  0.01027 -0.03509
3 -0.50920 -0.78545  0.04724 -0.13333  0.05556  0.33333 -0.00000
4 -0.34940  0.11111  0.50000  0.00000  0.00000  0.00000      NaN
5 -0.44828  2.04167  0.66667      NaN      NaN      NaN      NaN
6  0.84615 -0.16327 -0.50000      NaN      NaN      NaN      NaN
--------------------ARMA (0,2): y(t) = e(t) + 0.5e(t-1) - 0.4e(t-2)--------------------
Enter the number of data samples :>? 10000
Enter AR order:>? 0
Enter MA order:>? 2
Enter Mean of white noise:>? 1
Enter variance of white noise:>? 2
```

```
coefficient of MA b1>? 0.5
coefficient of MA b2>? -0.4
Is this stationary process:  True
GPAC Table for y:
          1        2        3        4        5        6        7
0  0.21800 -0.35751  0.20422 -0.17941  0.14310 -0.12807  0.11203
1 -1.34404 -0.24891 -0.09679 -0.02177 -0.01411 -0.00490  0.01248
2 -0.01706 -0.09014 -0.01407  0.03252 -0.01020 -0.04348  0.02632
3  5.20000 -0.08901 -0.18182  0.00000 -0.00000 -0.00000  0.00000
4  0.00000 -0.30882 -0.00000      NaN      NaN      NaN      NaN
5     -inf -0.28571      NaN      NaN      NaN      NaN      NaN
6 -0.25000 -1.33333      NaN      NaN      NaN      NaN      NaN
--------------------ARMA (2,2): y(t)+0.5y(t-1) +0.2y(t-2) = e(t)+0.5e(t-1) - 0.4e(t-2)--------------------
Enter the number of data samples :>? 10000
Enter AR order:>? 2
Enter MA order:>? 2
Enter Mean of white noise:>? 1
Enter variance of white noise:>? 2
Enter coefficient of AR a1>? 0.5
Enter coefficient of AR a2>? 0.2
coefficient of MA b1>? 0.5
coefficient of MA b2>? -0.4
Is this stationary process:  True
GPAC Table for y:
          1        2        3        4        5        6        7
0 -0.12800 -0.43145  0.13119 -0.19660  0.11402 -0.12972  0.09517
1  3.18750 -0.46312 -0.50426 -0.12349 -0.10670 -0.04739 -0.04912
2 -0.57598 -0.20830  0.04492  0.03856 -0.00000  0.02326  0.00833
3 -0.14894 -0.27675  0.23932  0.03448      NaN  0.00000  0.00000
4  1.22857 -0.23389  0.03571  0.50000      NaN      NaN      NaN
5 -0.53488 -0.20000  1.50000  0.00000      NaN      NaN
```

File - unknown

```
180       NaN
181 6  0.00000 -0.64151  0.33333      NaN      NaN      NaN
          NaN
182
```

Page 7 of 7

1.

Develop a python code that generates ARMA (na,nb) process.

The program should be written in a way that askes a user to enter the following information.

Hint: Use statsemodels library.

a. Enter the number of data samples: _______________

b. Enter the mean of white noise: ______________

c. Enter the variance of the white noise: ______________

d. Enter AR order: _______________

e. Enter MA order: ______________

f. Enter the coefficients of AR (you need to include a hint how this should be entered):____

g. Enter the coefficients of MA (you need to include a hint how this should be entered):____

```
Enter the number of data samples :>? 1000
Enter AR order:>? 1
Enter MA order:>? 0
Enter Mean of white noise:>? 1
Enter variance of white noise:>? 2
Enter coefficient of AR a1>? -0.5
Is this stationary process:  True
```

2.

Edit the python code in step 1 that implement the GPAC table using the following equation.

The output should be the GPAC table

GPAC Table for y:

	1	2	3	4	5	6	7
0	0.44600	-0.03860	0.07675	-0.00459	-0.04416	0.02289	0.00333
1	0.37668	0.84702	0.07442	-0.74359	-0.04639	0.02989	0.32941
2	0.72619	0.08706	0.32787	0.00575	-0.02410	-0.13636	-0.10714
3	0.61475	-2.35965	0.33333	1.00000	-0.00000	-0.00000	-0.00000
4	0.02667	0.08364	0.20000	-1.00000	NaN	NaN	NaN
5	3.00000	0.02222	0.37500	-1.00000	NaN	NaN	NaN
6	2.33333	-39.00000	0.33333	-1.00000	NaN	NaN	NaN

3.

Using the developed code above, simulate ARMA(1,0) for 1000 samples as follows:

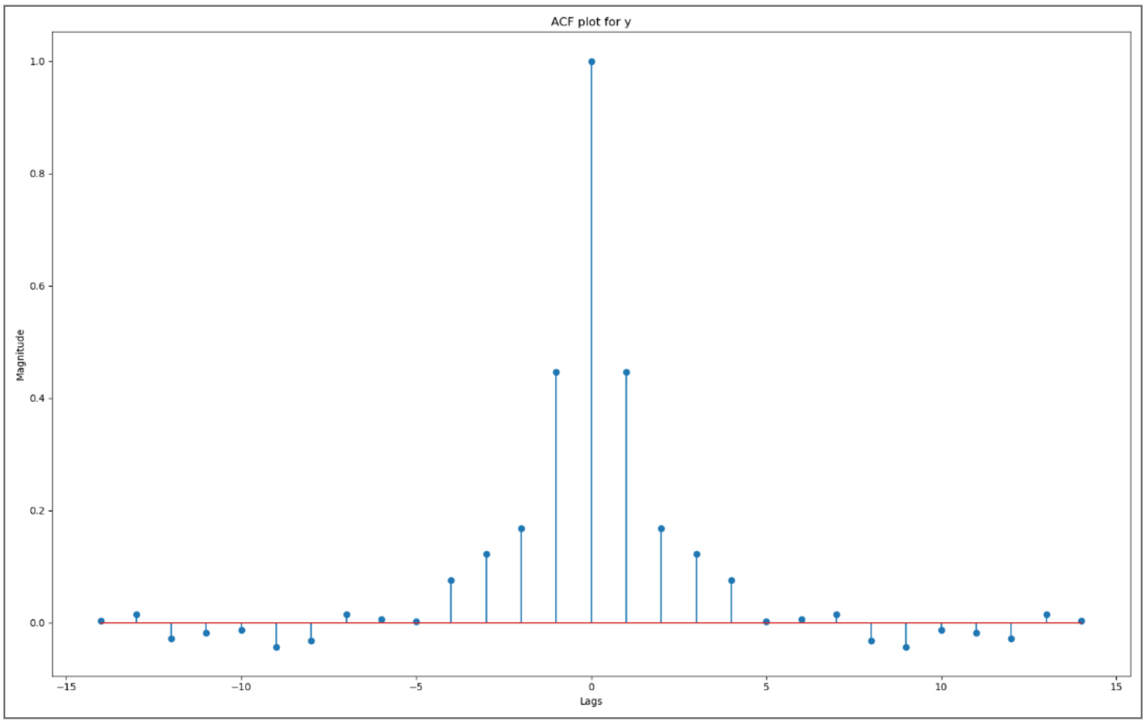
$y(t) - 0.5y(t-1) = e(t)$

Use statsemodels library to simulate above ARMA (1,2) process.

You can use the .generate_sample (# of samples, scales = std of WN noise) + mean (y) where mean(y) = μe(1+∑ bi) /1+∑ ai

4.

Using python program, estimate ACF for y(t) using the ACF function developed in previous labs. Number of lags = 15.



5.

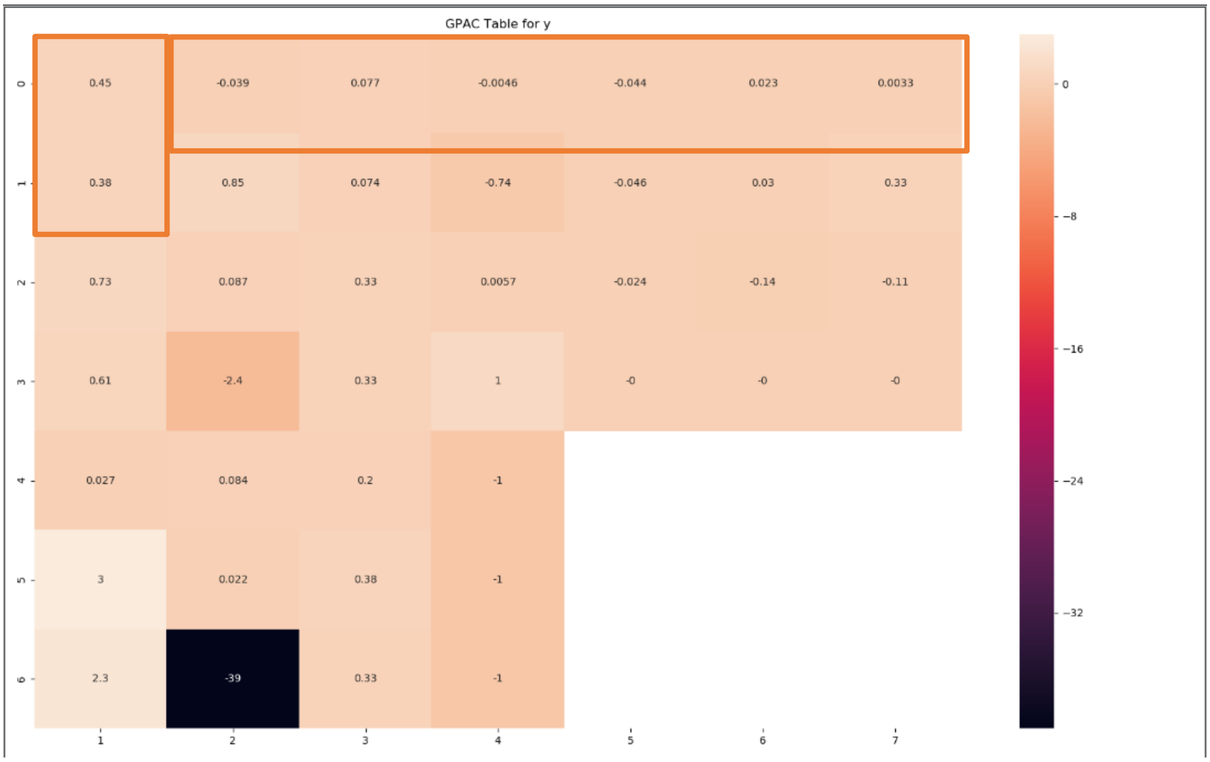
Using the estimated ACF from previous question, display GPAC table for k=7 and j=7.

Do you see a pattern of constant column 0.5 and a row of zeros?

What is the estimated na and what is the estimated nb?

Yes I see a pattern of constant column 0.5 and a row of zeros

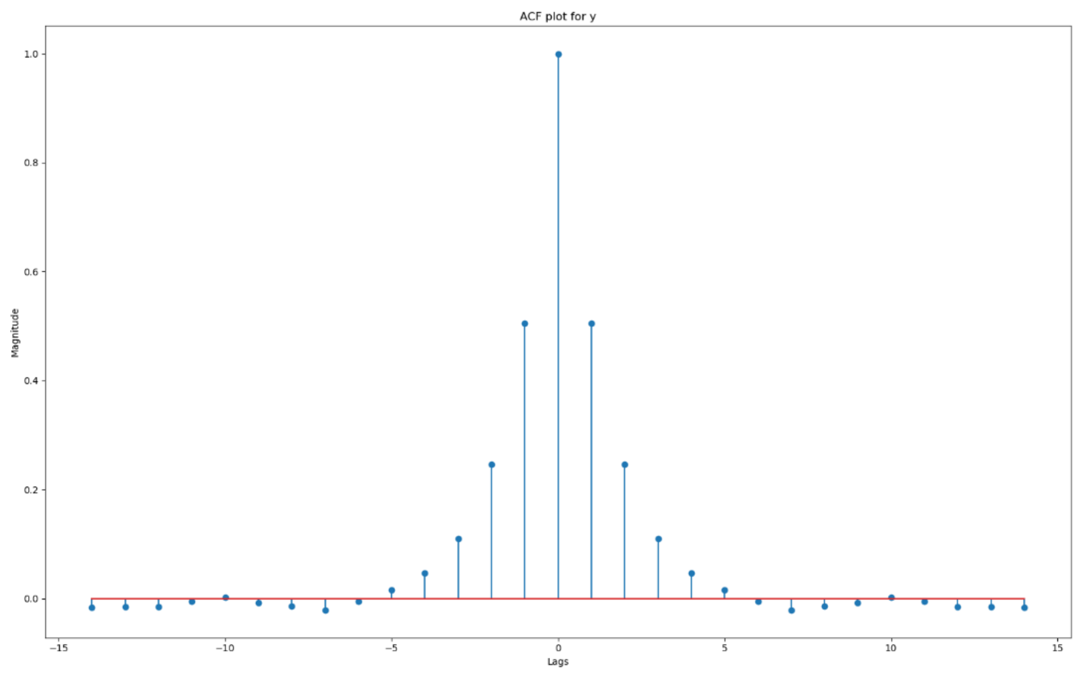
Estimated na : 1; nb: 0

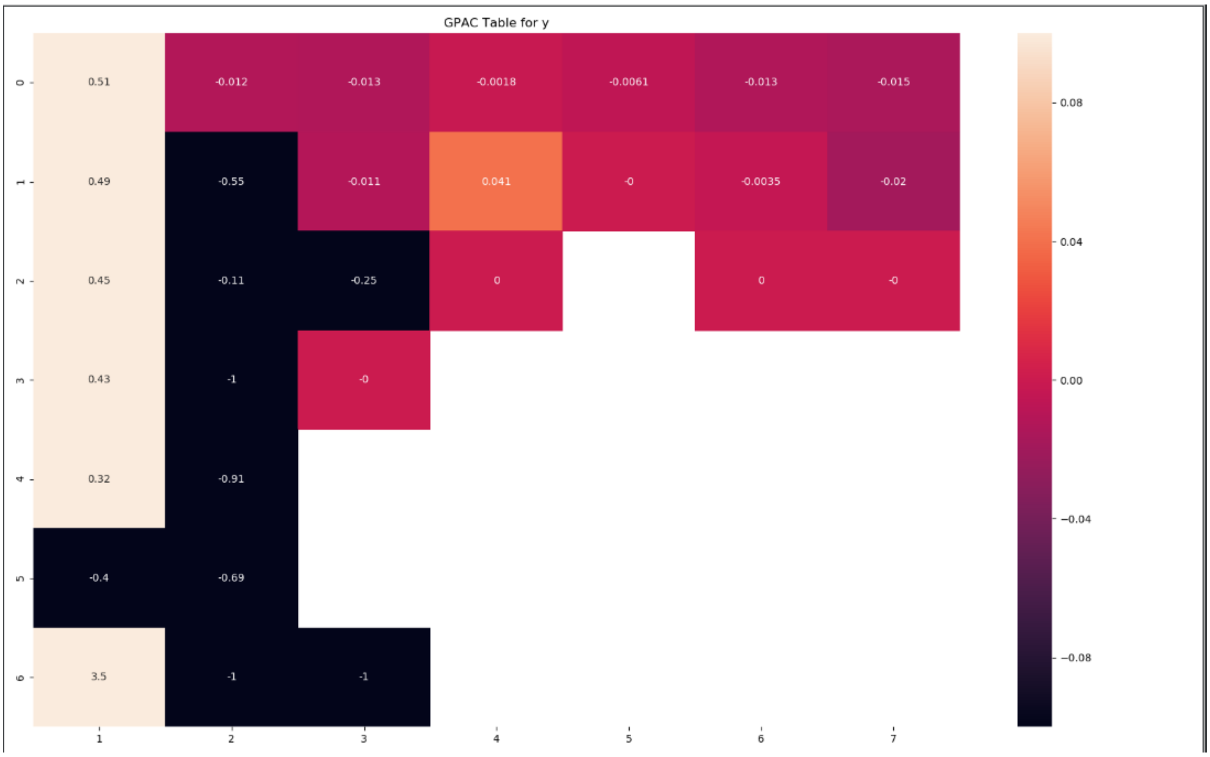


6. Increase the number of samples to 5000 and 10000. Do the numbers in the pattern converge?

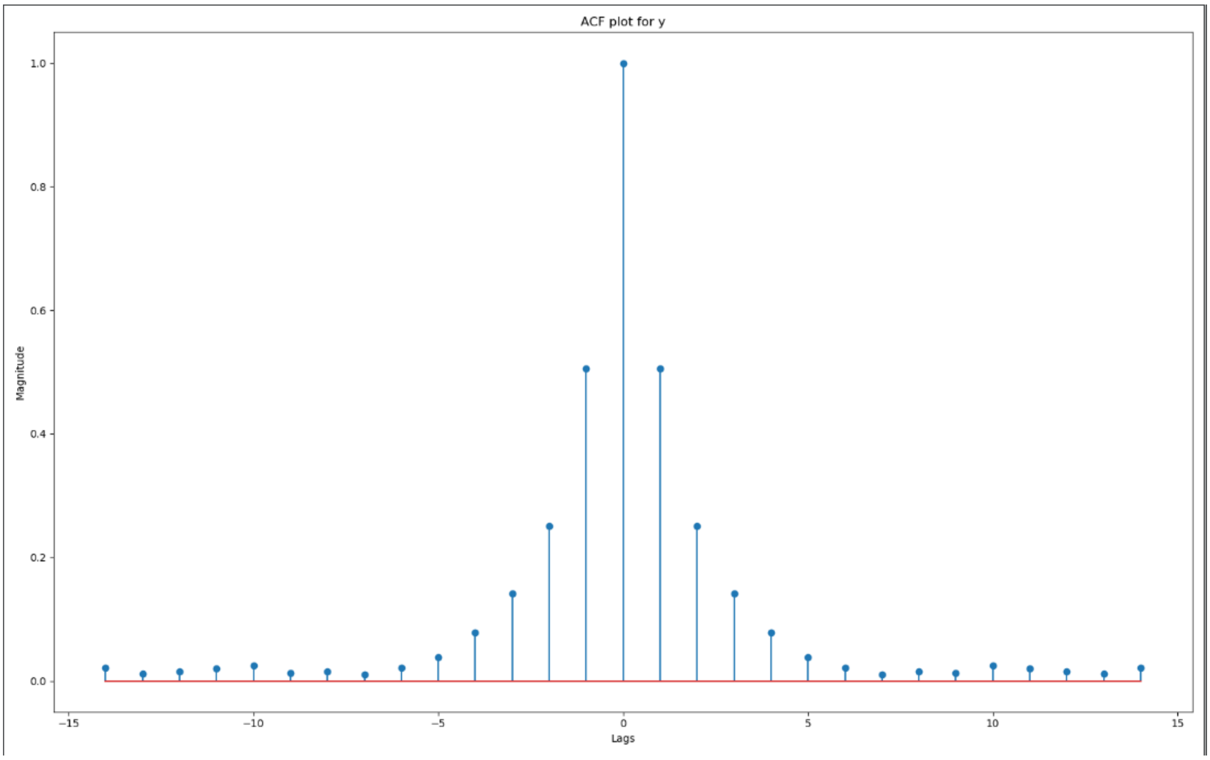
Yes the numbers converge clear distinction is seen in below heatmap

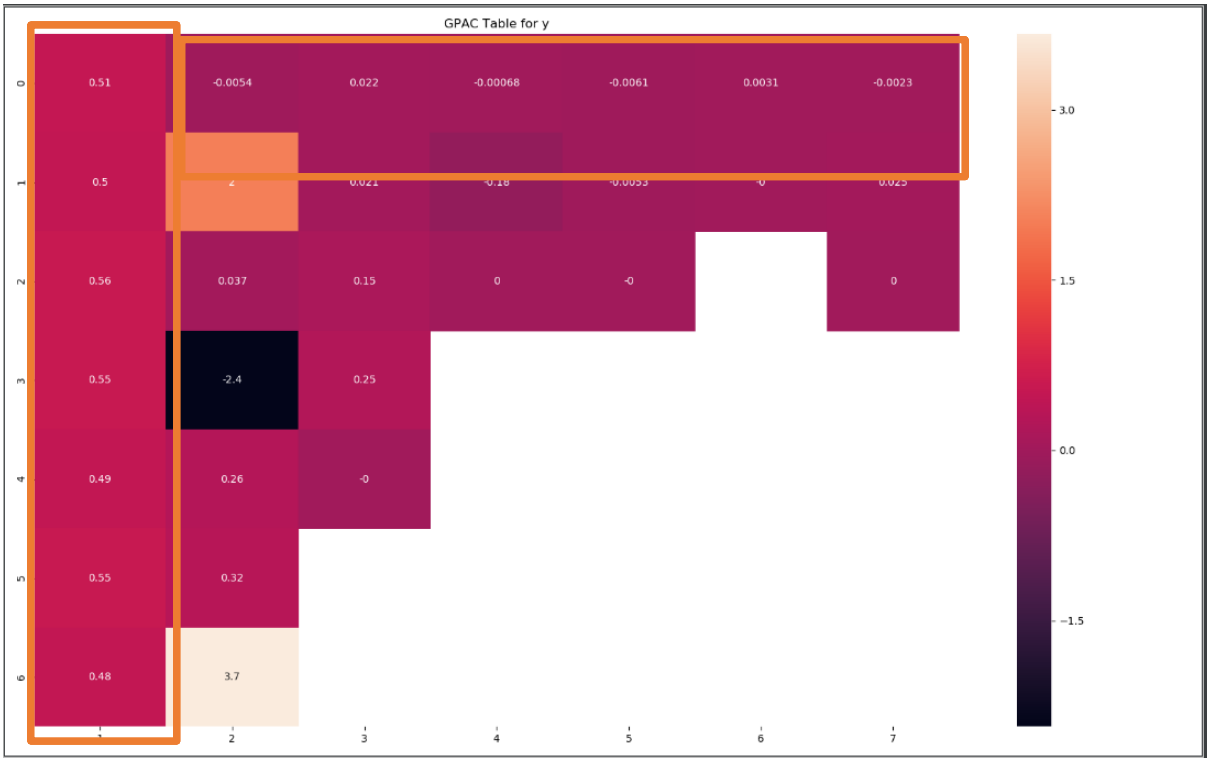
Samples = 5000





Samples 10000





7. Repeat above steps for the following 7 examples with 10000 samples which should cover the followings:

a. Display the GPAC table (k=7, j=7) in your report and highlight the pattern that read the process orders. Assign the correct label for each column (i.e. k=1, k=2, …) and each row (i.e. j=0, j=1, ...)

b. Display the ACF for the y(t) for 15 lags.

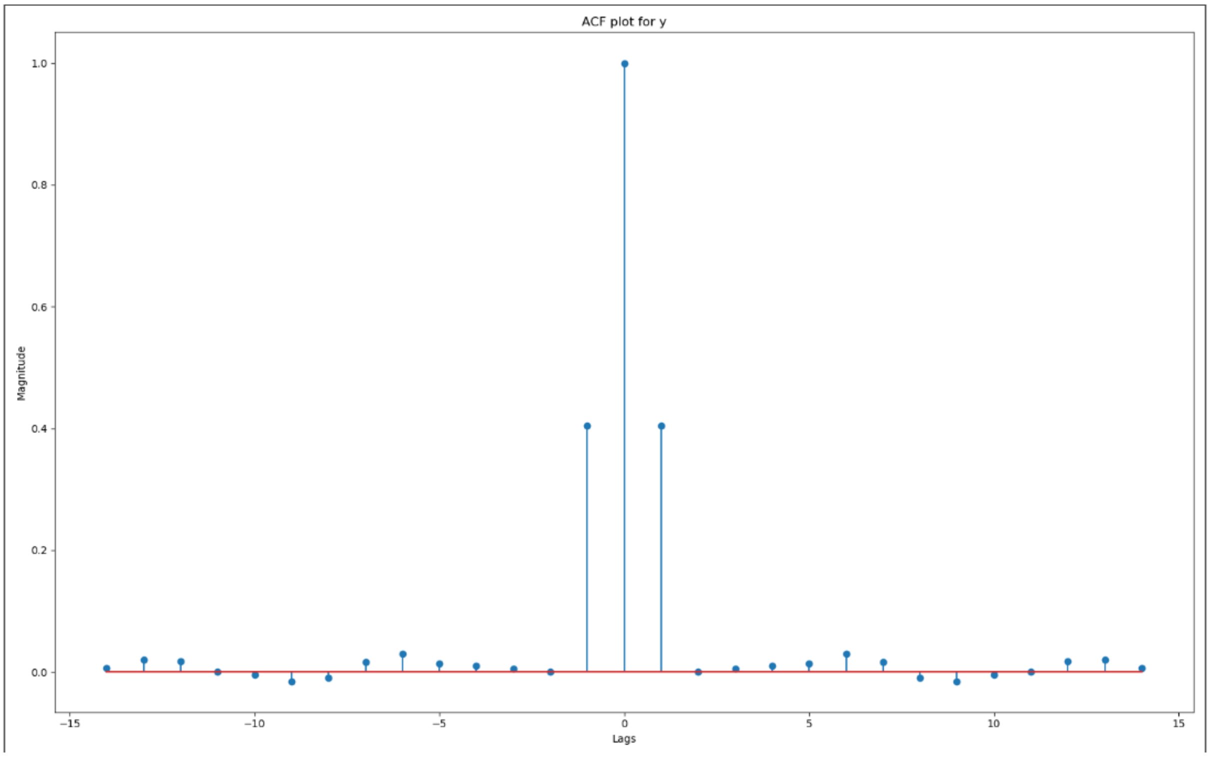
c. Based on the observed pattern in the GPAC table, what is the estimated na and estimated nb. Compare the estimated order versus the true order. Write down your observations.

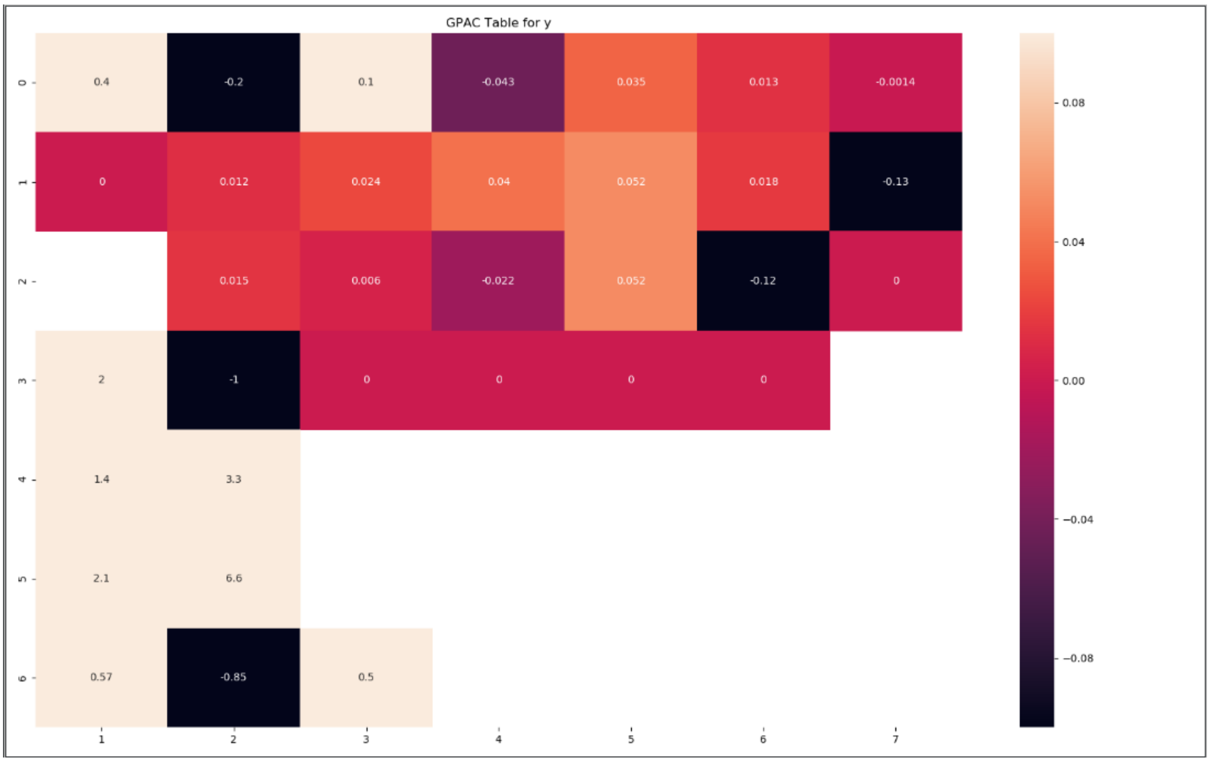
d. Make sure to include the .py code that can be run and verify the results.

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

Estimated na: 0 Estimated nb: 1 | True Orders na:0 nb: 1.

Estimated and true orders match.

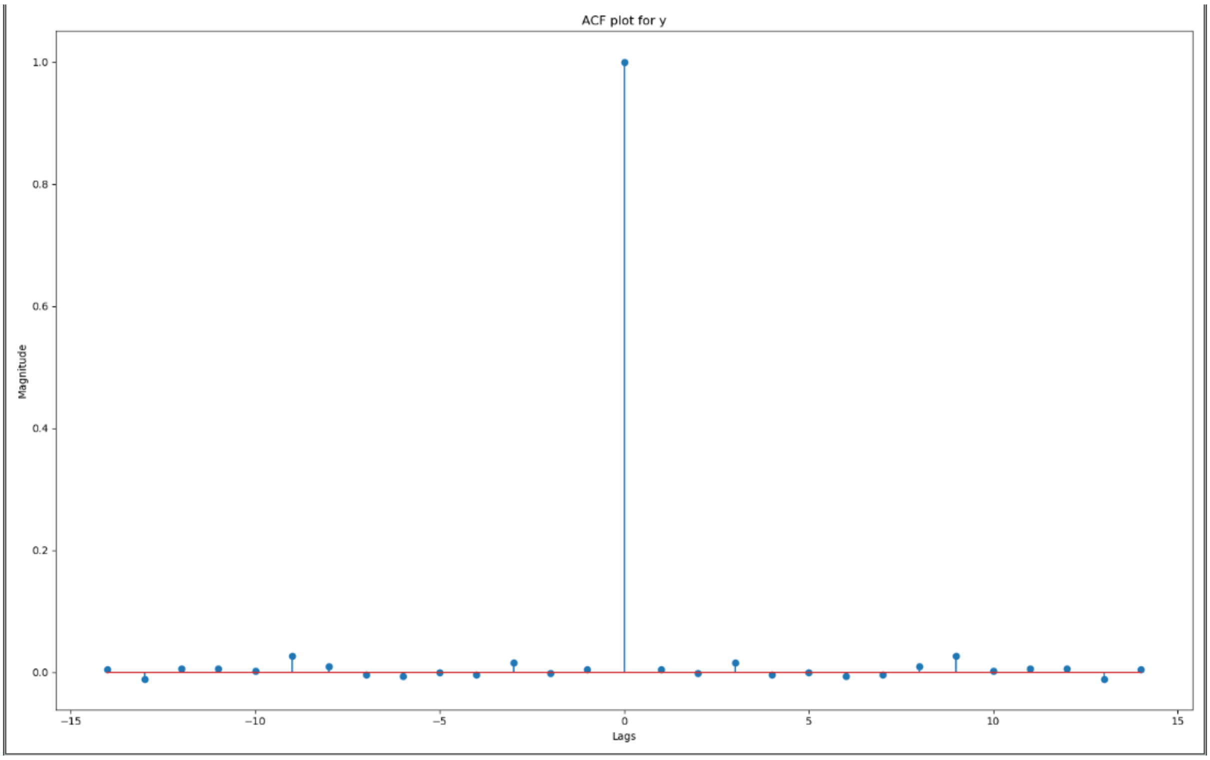


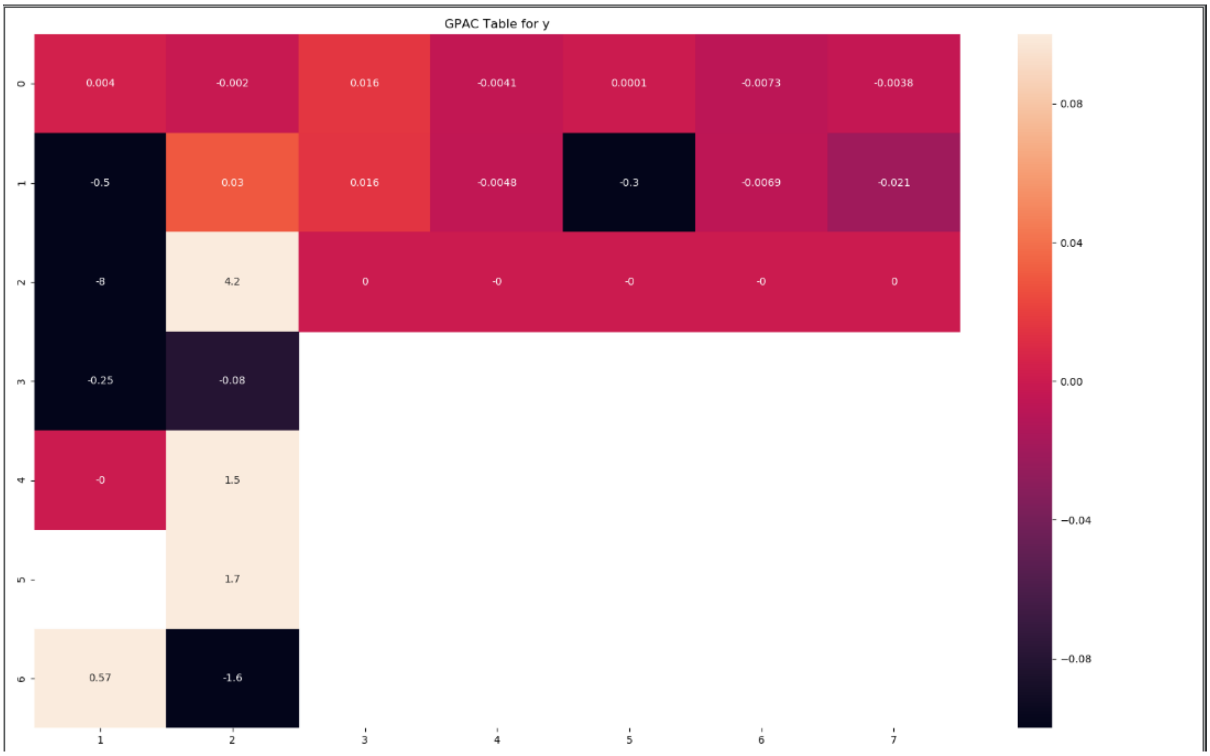


Example 3: ARMA (1,1): y(t) + 0.5y(t-1) + 0.2y(t-2) = e(t)

Estimated na: 1 Estimated nb: 1 | True Orders na: 1 nb: 1.

Estimated and true orders match.

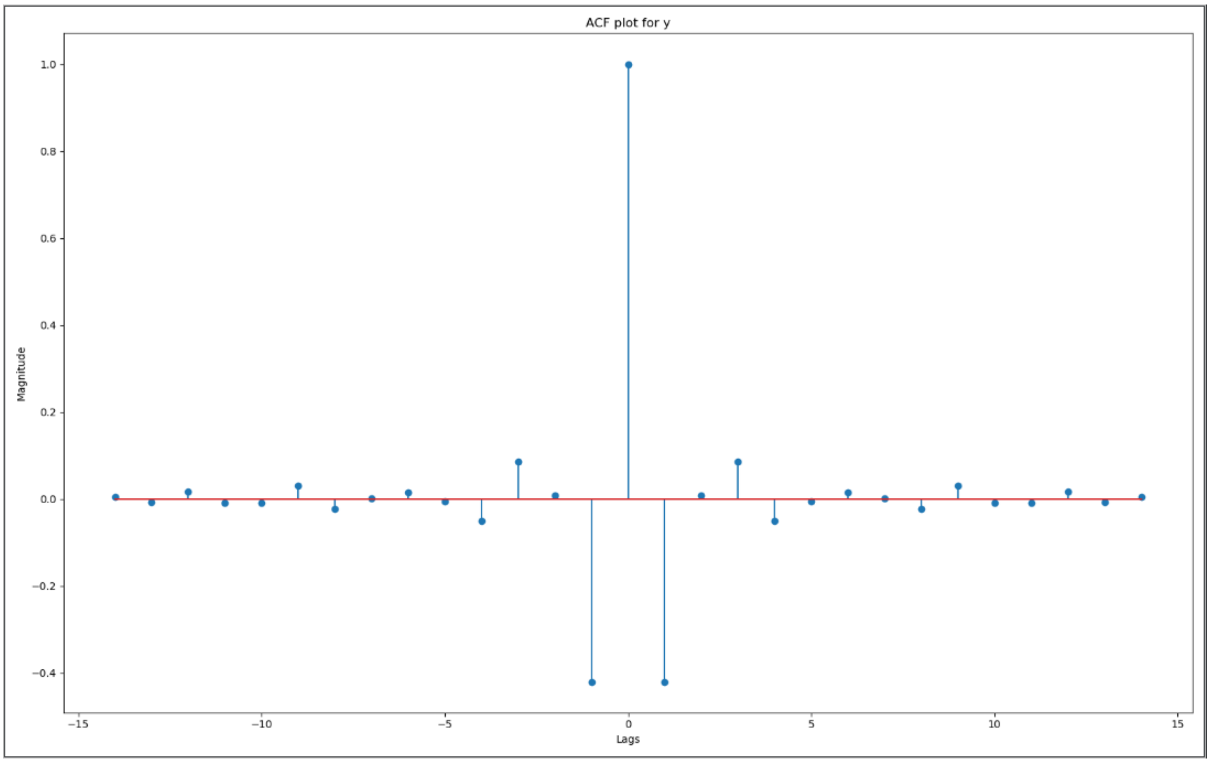


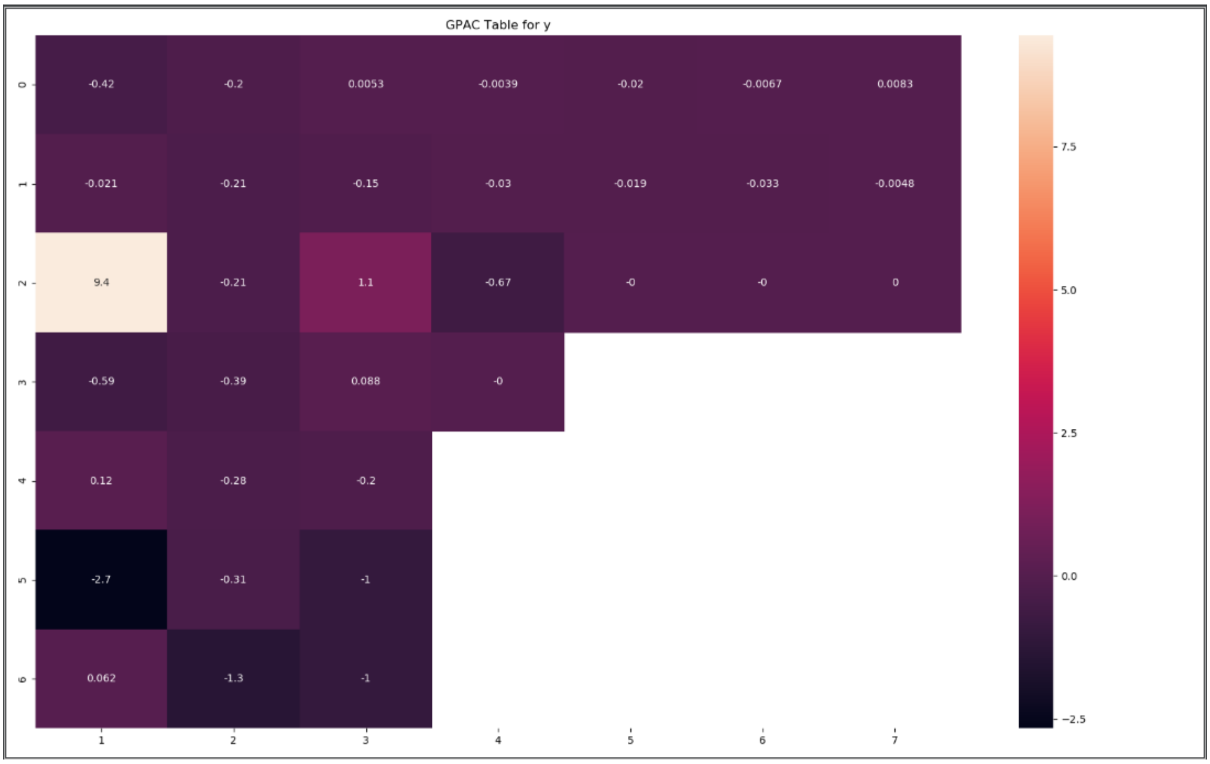


Example 4: ARMA (2,0): y(t) + 0.5y(t-1) + 0.2y(t-2) = e(t)

Estimated na: 2 Estimated nb: 0 | True Orders na: 2 nb: 0.

Estimated and true orders match.

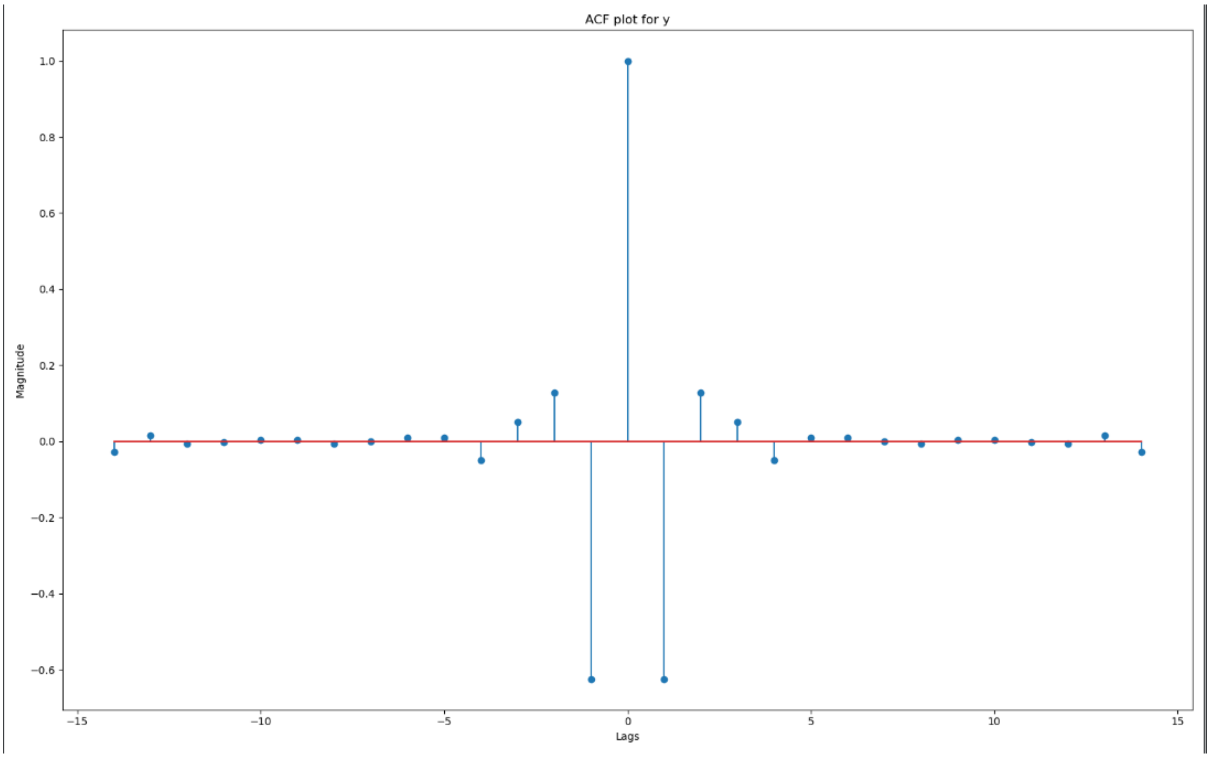


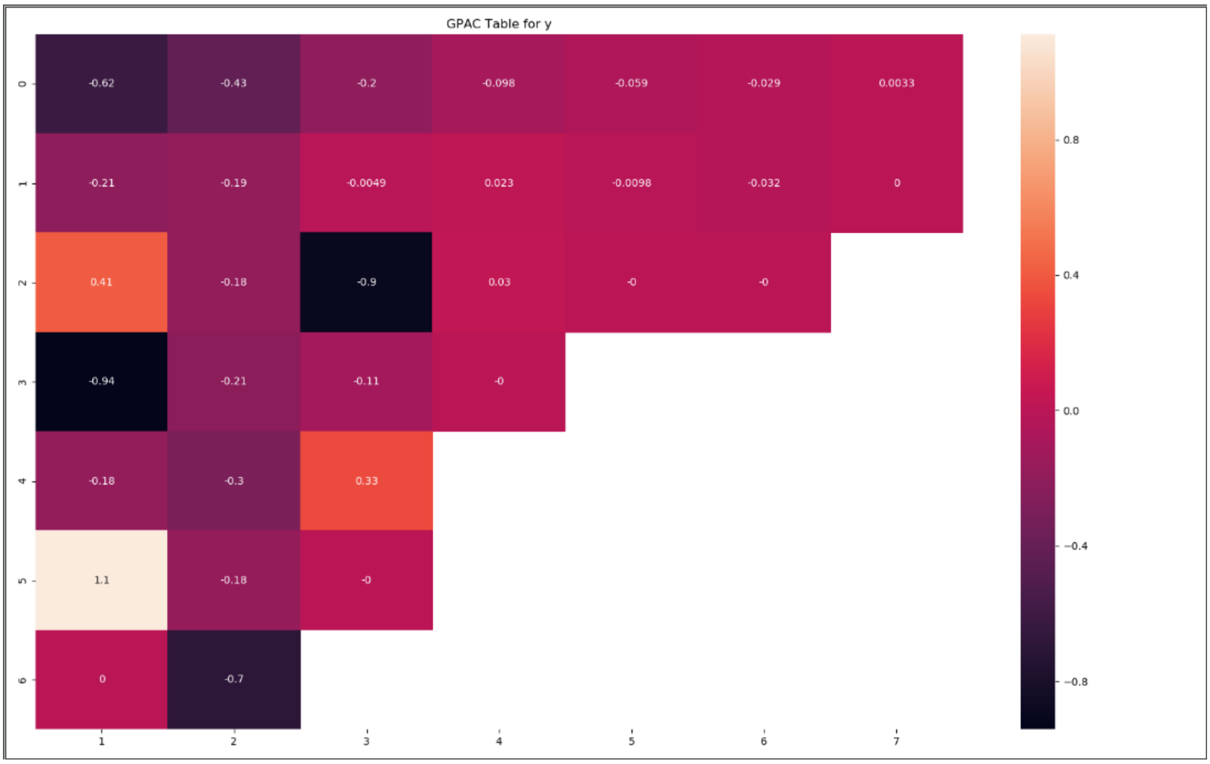


Example 5: ARMA (2,1): y(t) + 0.5y(t-1) + 0.2y(t-2) = e(t) - 0.5e(t-1)

Estimated na: 2 Estimated nb: 1 | True Orders na: 2 nb: 1.

Estimated and true orders match.

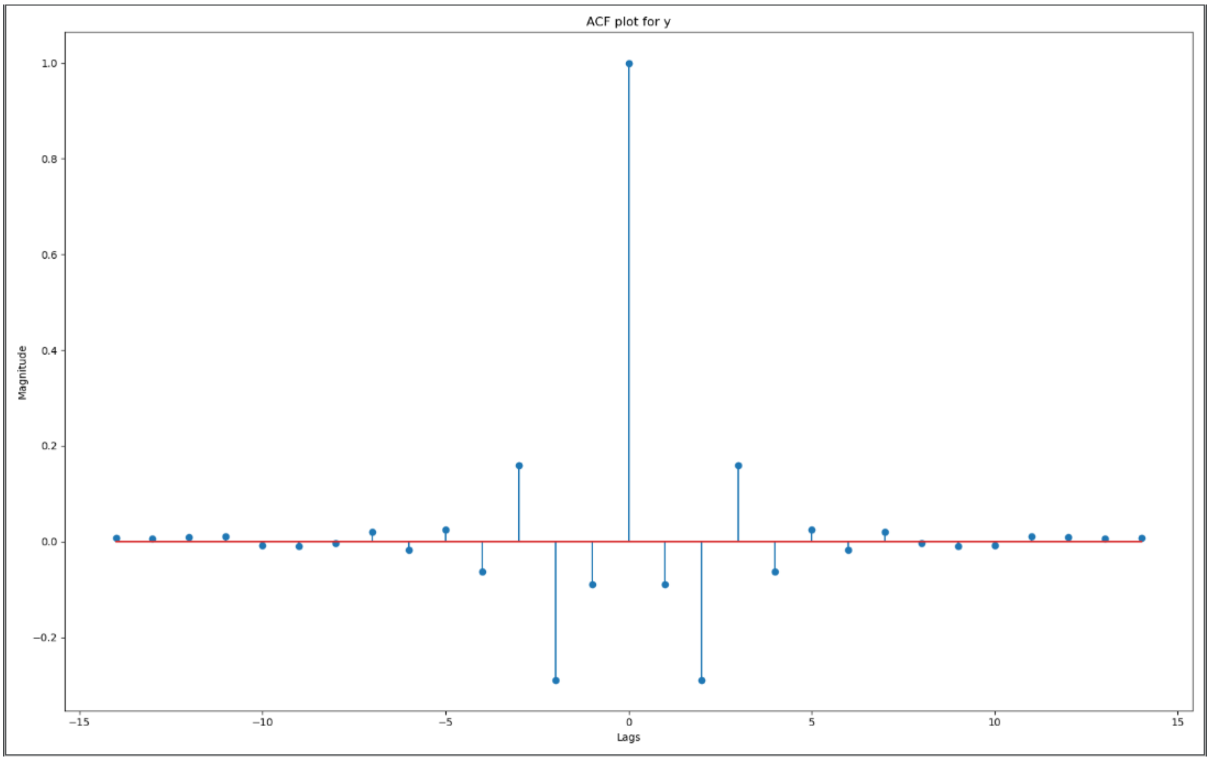


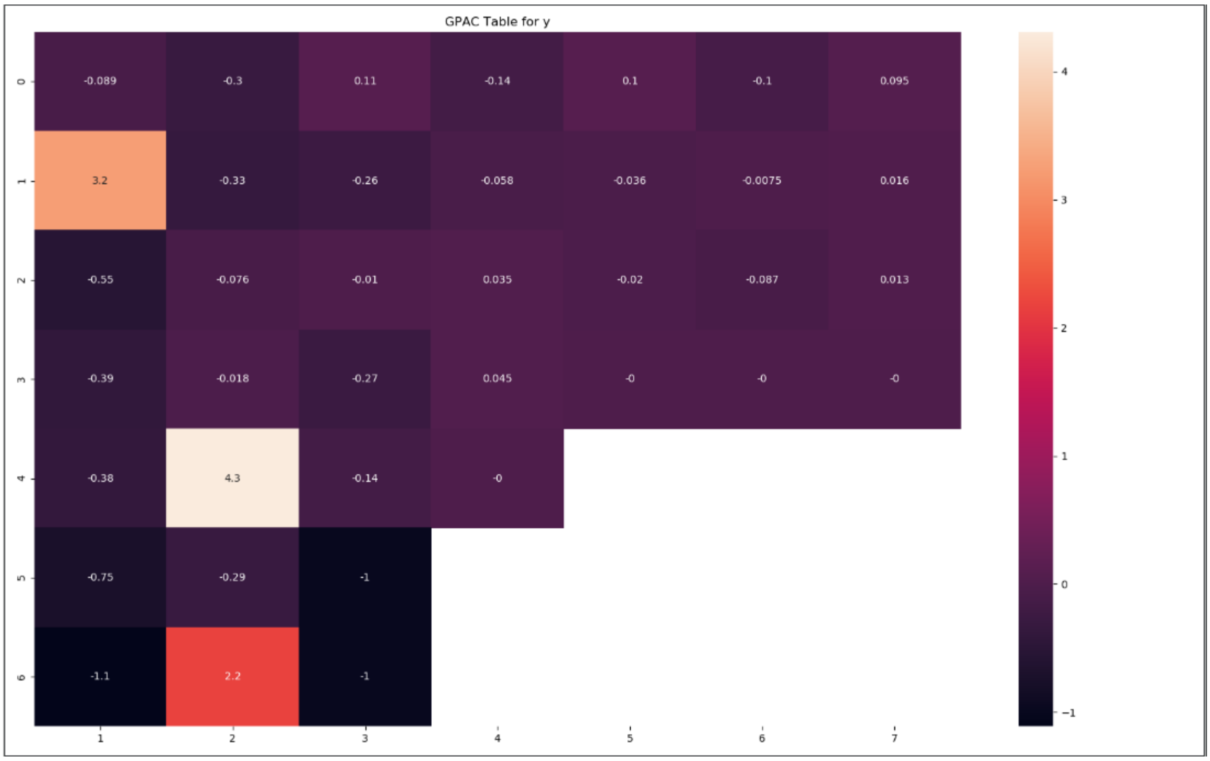


Example 6: ARMA (1,2): y(t) + 0.5y(t-1) = e(t) + 0.5e(t-1) - 0.4e(t-2)

Estimated na: 1 Estimated nb: 2 | True Orders na: 1 nb: 2

Estimated and true orders match.

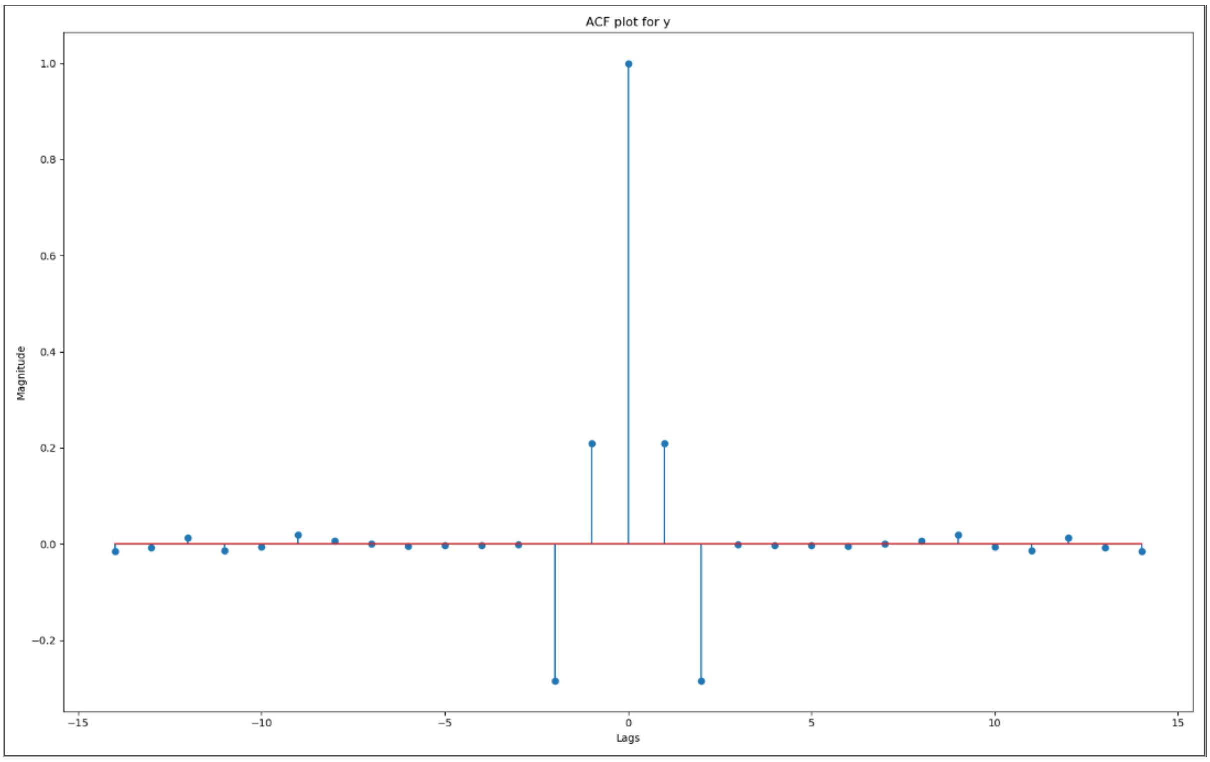


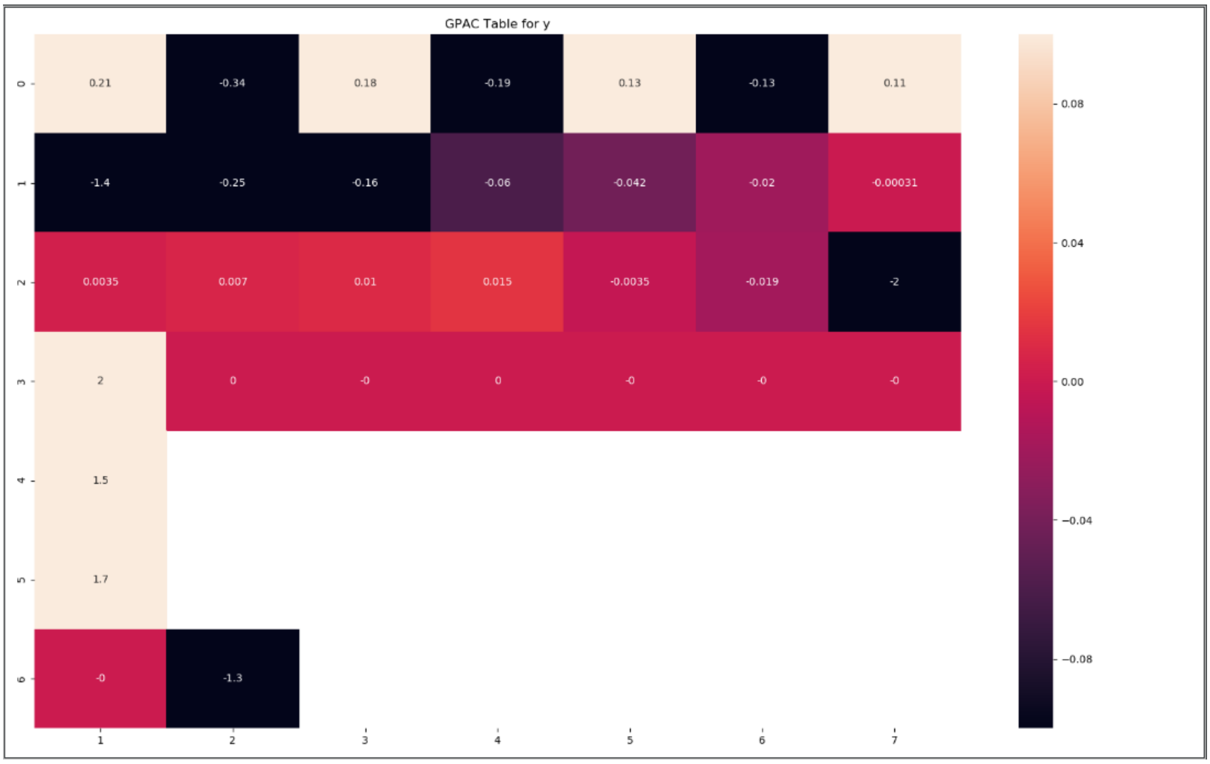


Example 7: ARMA (0,2): y(t) = e(t) + 0.5e(t-1) - 0.4e(t-2)

Estimated na: 0 Estimated nb: 2 | True Orders na: 0 nb: 2

Estimated and true orders match.

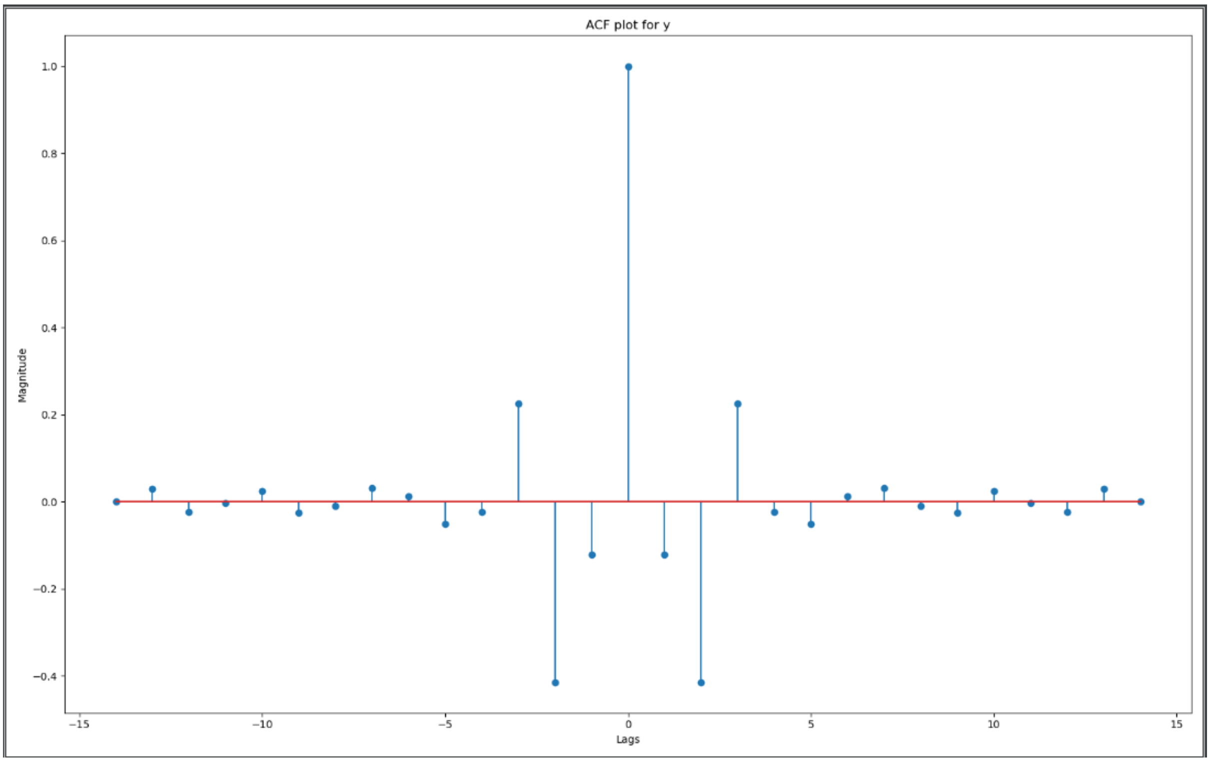


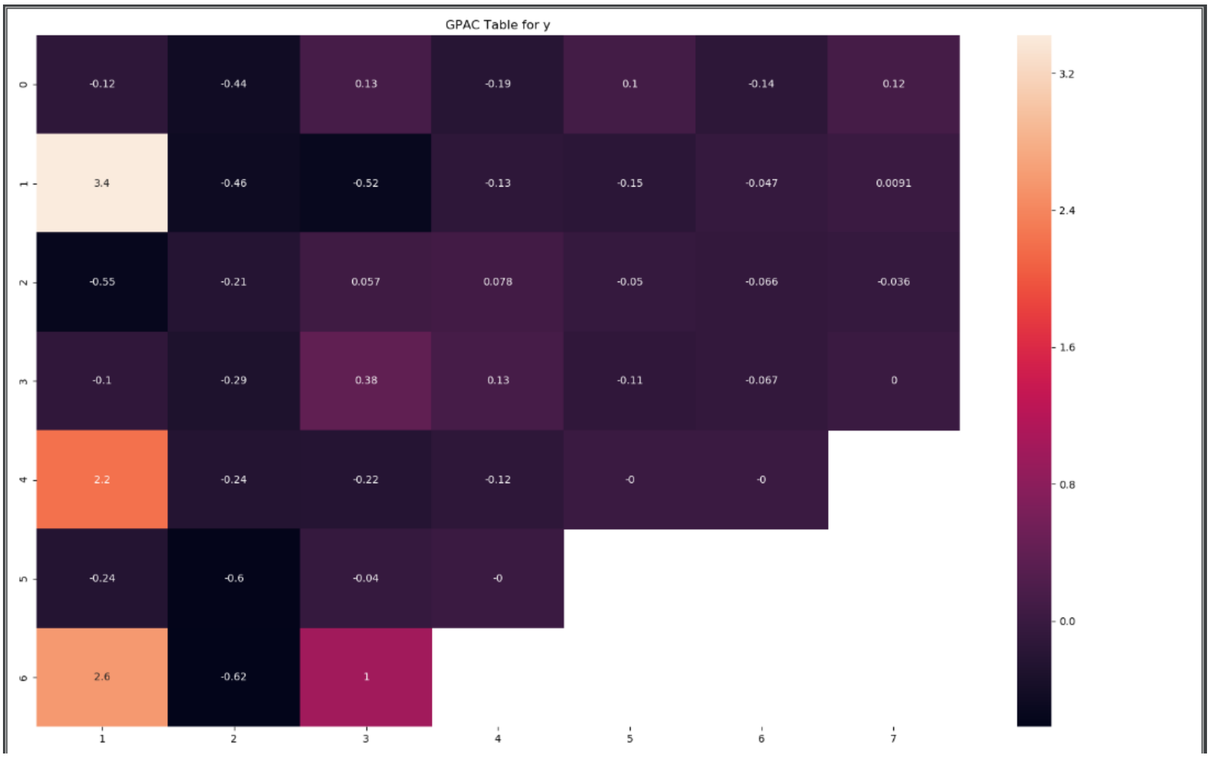


Example 8: ARMA (0,2): y(t) = e(t) + 0.5e(t-1) - 0.4e(t-2)

Estimated na: 0 Estimated nb: 2 | True Orders na: 0 nb: 2

Estimated and true orders match.





CONCLUSION

In this LAB we implemented the GPAC array covered in lecture using Python program and tested the accuracy of code using an ARMA(na,nb) model. Using statsmodels we simulated the ARMA process and GPAC code that generates GPAC table for various numbers of rows and columns was Coded. Thus we tested the above code for various equations and made a comparison with estimated orders and true orders and ACF were plotted.

CHALLENGE

GPAC Calculations was little tricky to understand in the beginning.

APPENDIX

```
import statsmodels.api as sm
import numpy as np
import matplotlib.pyplot as plt
from functions import *
import warnings
warnings.filterwarnings("ignore")

#%%=================================================================================
#1 . y(t) − 0.5y(t − 1) = e(t)
# %%--------------------------------------------------------------------------------
print(20 * "-" + "y(t) − 0.5y(t − 1) = e(t)" + 20 * "-")
lags = 15
T = int(input("Enter the number of data samples :"))
na = int(input("Enter AR order:"))
nb = int(input("Enter MA order:"))
mean = int(input("Enter Mean of white noise:"))
std = int(input("Enter variance of white noise:"))
scales = np.sqrt(std)
mean_y = (1*(1+1))/((1-0.5))

an = [0]*na
bn = [0]*nb

for i in range(na):
    an[i] = float(input("Enter coefficient of AR a{}".format(i+1)))
    # an_sum = np.sum(an[i])
    # print("Sum of an[i] = ",an_sum)

for i in range(nb):
    bn[i] = float(input("coefficient of MA b{}".format(i+1)))
    # bn_sum = np.sum(bn[i])
    # print("Sum of bn[i] = ",bn_sum)

max_order = max(na,nb)
num = [0]*(max_order+1)
den = [0]*(max_order+1)
for i in range(na+1):
    if i==0:
        den[i] = 1
    else:
        den[i] = an[i-1]

arparmas = np.array(an)
maparams = np.array(bn)

na = len(arparmas)
nb = len(maparams)
```

```
ar = np.r_[1, arparmas]
ma = np.r_[1, maparams]

arma_process = sm.tsa.ArmaProcess(ar,ma)
print("Is this stationary process: ", arma_process.isstationary)

y = arma_process.generate_sample(T, scales) + mean_y
# y = y + mean_y

# ACF of the dependent variable.
autocorrelation = cal_auto_correlation(y, 15)
plot_acf(autocorrelation, "ACF plot for y")
#gpac
j = 7
k = 7
lags = j + k

# autocorrelation of traffic volume
ry = cal_auto_correlation(y, lags)

# create GPAC Table
gpac_table = create_gpac_table(j, k, ry)
print("GPAC Table for y:")
print(gpac_table.to_string())

# heatmap
plot_heatmap(gpac_table, "GPAC Table for y")

# %%------------------------------------------------------------------
print(" Repeat for 5000 samples")
T1 = int(input("Enter the number of data samples:"))
y1 = arma_process.generate_sample(T1, scales) + mean_y

# ACF of the dependent variable.
autocorrelation = cal_auto_correlation(y1, 15)
plot_acf(autocorrelation, "ACF plot for y")

# autocorrelation of traffic volume
ry = cal_auto_correlation(y1, lags)

# create GPAC Table
gpac_table = create_gpac_table(j, k, ry)
print("GPAC Table for y:")
print(gpac_table.to_string())

# heatmap
plot_heatmap(gpac_table, "GPAC Table for y")
# %%------------------------------------------------------------------
------------------------------
print(" Repeat for 10000 samples")
T2 = int(input("Enter the number of data samples:"))
y2 = arma_process.generate_sample(T2, scales) + mean_y

# ACF of the dependent variable.
autocorrelation = cal_auto_correlation(y2, 15)
plot_acf(autocorrelation, "ACF plot for y")

# autocorrelation of traffic volume
```

```python
ry = cal_auto_correlation(y2, lags)

# create GPAC Table
gpac_table = create_gpac_table(j, k, ry)
print("GPAC Table for y:")
print(gpac_table.to_string())

# heatmap
plot_heatmap(gpac_table, "GPAC Table for y")
print()

#%%=================================================================================
#2 . y(t) = e(t) + 0.5e(t-1)
# %%--------------------------------------------------------------------------------
print(20 * "-" + "y(t) = e(t) + 0.5e(t-1)" + 20 * "-")
T = int(input("Enter the number of data samples :"))
na = int(input("Enter AR order:"))
nb = int(input("Enter MA order:"))
mean = int(input("Enter Mean of white noise:"))
std = int(input("Enter variance of white noise:"))
scales = np.sqrt(std)
mean_y2 = (1*(1+0.5))/((1))

an = [0]*na
bn = [0]*nb

for i in range(na):
    an[i] = float(input("Enter coefficient of AR a{}".format(i+1)))
    # an_sum = np.sum(an[i])
    # print("Sum of an[i] = ",an_sum)

for i in range(nb):
    bn[i] = float(input("coefficient of MA b{}".format(i+1)))
    # bn_sum = np.sum(bn[i])
    # print("Sum of bn[i] = ",bn_sum)

max_order = max(na,nb)
num = [0]*(max_order+1)
den = [0]*(max_order+1)
for i in range(na+1):
    if i==0:
        den[i] = 1
    else:
        den[i] = an[i-1]

arparmas = np.array(an)
maparams = np.array(bn)

na = len(arparmas)
nb = len(maparams)

ar = np.r_[1, arparmas]
ma = np.r_[1, maparams]

arma_process = sm.tsa.ArmaProcess(ar,ma)
print("Is this stationary process: ", arma_process.isstationary)

y = arma_process.generate_sample(T, scales) + mean_y2
```

```python
# ACF of the dependent variable.
autocorrelation = cal_auto_correlation(y, 15)
plot_acf(autocorrelation, "ACF plot for y")

#gpac
j = 7
k = 7
lags = j + k

# autocorrelation of traffic volume
ry = cal_auto_correlation(y, lags)

# create GPAC Table
gpac_table = create_gpac_table(j, k, ry)
print("GPAC Table for y:")
print(gpac_table.to_string())

# heatmap
plot_heatmap(gpac_table, "GPAC Table for y")

#
%%=================================================================================
============================
#3 . ARMA (1,1): y(t) + 0.5y(t-1) = e(t) + 0.5e(t-1)
# %%-------------------------------------------------------------------------------
------------------------------
print(20 * "-" + "ARMA (1,1): y(t) + 0.5y(t-1) = e(t) + 0.5e(t-1)" + 20 * "-")
T = int(input("Enter the number of data samples :"))
na = int(input("Enter AR order:"))
nb = int(input("Enter MA order:"))
mean = int(input("Enter Mean of white noise:"))
std = int(input("Enter variance of white noise:"))
scales = np.sqrt(std)
mean_y3 = (1*(1+0.5))/((1+0.5))

an = [0]*na
bn = [0]*nb

for i in range(na):
    an[i] = float(input("Enter coefficient of AR a{}".format(i+1)))

for i in range(nb):
    bn[i] = float(input("coefficient of MA b{}".format(i+1)))

max_order = max(na,nb)
num = [0]*(max_order+1)
den = [0]*(max_order+1)
for i in range(na+1):
    if i==0:
        den[i] = 1
    else:
        den[i] = an[i-1]

arparmas = np.array(an)
maparams = np.array(bn)

na = len(arparmas)
```

```
nb = len(maparams)

ar = np.r_[1, arparmas]
ma = np.r_[1, maparams]

arma_process = sm.tsa.ArmaProcess(ar,ma)
print("Is this stationary process: ", arma_process.isstationary)

y = arma_process.generate_sample(T, scales) + mean_y3

# ACF of the dependent variable.
autocorrelation = cal_auto_correlation(y, 15)
plot_acf(autocorrelation, "ACF plot for y")

#gpac
j = 7
k = 7
lags = j + k

# autocorrelation
ry = cal_auto_correlation(y, lags)

# create GPAC Table
gpac_table = create_gpac_table(j, k, ry)
print("GPAC Table for y:")
print(gpac_table.to_string())

# heatmap
plot_heatmap(gpac_table, "GPAC Table for y")

#
%%==================================================================================
============================
#4 . ARMA (2,0): y(t) + 0.5y(t-1) + 0.2y(t-2) = e(t)
# %%--------------------------------------------------------------------------------
------------------------------
print(20 * "-" + "ARMA (2,0): y(t) + 0.5y(t-1) + 0.2y(t-2) = e(t)" + 20 * "-")
T = int(input("Enter the number of data samples :"))
na = int(input("Enter AR order:"))
nb = int(input("Enter MA order:"))
mean = int(input("Enter Mean of white noise:"))
std = int(input("Enter variance of white noise:"))
scales = np.sqrt(std)
mean_y4 = (1*(1))/((1+0.5+0.2))

an = [0]*na
bn = [0]*nb

for i in range(na):
    an[i] = float(input("Enter coefficient of AR a{}".format(i+1)))

for i in range(nb):
    bn[i] = float(input("coefficient of MA b{}".format(i+1)))

max_order = max(na,nb)
num = [0]*(max_order+1)
den = [0]*(max_order+1)
for i in range(na+1):
```

```
    if i==0:
        den[i] = 1
    else:
        den[i] = an[i-1]

arparmas = np.array(an)
maparams = np.array(bn)

na = len(arparmas)
nb = len(maparams)

ar = np.r_[1, arparmas]
ma = np.r_[1, maparams]

arma_process = sm.tsa.ArmaProcess(ar,ma)
print("Is this stationary process: ", arma_process.isstationary)

y = arma_process.generate_sample(T, scales) + mean_y4

# ACF of the dependent variable.
autocorrelation = cal_auto_correlation(y, 15)
plot_acf(autocorrelation, "ACF plot for y")

#gpac
j = 7
k = 7
lags = j + k

# autocorrelation
ry = cal_auto_correlation(y, lags)

# create GPAC Table
gpac_table = create_gpac_table(j, k, ry)
print("GPAC Table for y:")
print(gpac_table.to_string())

# heatmap
plot_heatmap(gpac_table, "GPAC Table for y")
#
%%=================================================================================
============================
#5 . ARMA (2,1): y(t) + 0.5y(t-1) + 0.2y(t-2) = e(t) - 0.5e(t-1)
# %%-------------------------------------------------------------------------------
------------------------------
print(20 * "-" + "ARMA (2,1): y(t) + 0.5y(t-1) + 0.2y(t-2) = e(t) - 0.5e(t-1) " +
20 * "-")
T = int(input("Enter the number of data samples :"))
na = int(input("Enter AR order:"))
nb = int(input("Enter MA order:"))
mean = int(input("Enter Mean of white noise:"))
std = int(input("Enter variance of white noise:"))
scales = np.sqrt(std)
mean_y5 = (1*(1-0.5))/((1+0.5+0.2))

an = [0]*na
bn = [0]*nb

for i in range(na):
```

```
    an[i] = float(input("Enter coefficient of AR a{}".format(i+1)))

for i in range(nb):
    bn[i] = float(input("coefficient of MA b{}".format(i+1)))

max_order = max(na,nb)
num = [0]*(max_order+1)
den = [0]*(max_order+1)
for i in range(na+1):
    if i==0:
        den[i] = 1
    else:
        den[i] = an[i-1]

arparmas = np.array(an)
maparams = np.array(bn)

na = len(arparmas)
nb = len(maparams)

ar = np.r_[1, arparmas]
ma = np.r_[1, maparams]

arma_process = sm.tsa.ArmaProcess(ar,ma)
print("Is this stationary process: ", arma_process.isstationary)

y = arma_process.generate_sample(T, scales) + mean_y5

# ACF of the dependent variable.
autocorrelation = cal_auto_correlation(y, 15)
plot_acf(autocorrelation, "ACF plot for y")

#gpac
j = 7
k = 7
lags = j + k

# autocorrelation
ry = cal_auto_correlation(y, lags)

# create GPAC Table
gpac_table = create_gpac_table(j, k, ry)
print("GPAC Table for y:")
print(gpac_table.to_string())

# heatmap
plot_heatmap(gpac_table, "GPAC Table for y")

#
%%=================================================================================
============================
#6 . ARMA (1,2): y(t) + 0.5y(t-1) = e(t) + 0.5e(t-1) - 0.4e(t-2)
# %%-------------------------------------------------------------------------------
------------------------------
print(20 * "-" + "ARMA (1,2): y(t) + 0.5y(t-1) = e(t) + 0.5e(t-1) - 0.4e(t-2) " +
20 * "-")

T = int(input("Enter the number of data samples :"))
```

```
na = int(input("Enter AR order:"))
nb = int(input("Enter MA order:"))
mean = int(input("Enter Mean of white noise:"))
std = int(input("Enter variance of white noise:"))
scales = np.sqrt(std)
mean_y6 = (1*(1+0.5-0.4))/((1+0.5))

an = [0]*na
bn = [0]*nb

for i in range(na):
    an[i] = float(input("Enter coefficient of AR a{}".format(i+1)))

for i in range(nb):
    bn[i] = float(input("coefficient of MA b{}".format(i+1)))

max_order = max(na,nb)
num = [0]*(max_order+1)
den = [0]*(max_order+1)
for i in range(na+1):
    if i==0:
        den[i] = 1
    else:
        den[i] = an[i-1]

arparmas = np.array(an)
maparams = np.array(bn)

na = len(arparmas)
nb = len(maparams)

ar = np.r_[1, arparmas]
ma = np.r_[1, maparams]

arma_process = sm.tsa.ArmaProcess(ar,ma)
print("Is this stationary process: ", arma_process.isstationary)

y = arma_process.generate_sample(T, scales) + mean_y6

# ACF of the dependent variable.
autocorrelation = cal_auto_correlation(y, 15)
plot_acf(autocorrelation, "ACF plot for y")

#gpac
j = 7
k = 7
lags = j + k

# autocorrelation
ry = cal_auto_correlation(y, lags)

# create GPAC Table
gpac_table = create_gpac_table(j, k, ry)
print("GPAC Table for y:")
print(gpac_table.to_string())

# heatmap
plot_heatmap(gpac_table, "GPAC Table for y")
```

```
#%%==================================================================================
=============================
#7 .ARMA (0,2): y(t) = e(t) + 0.5e(t-1) - 0.4e(t-2)
# %%---------------------------------------------------------------------------------
------------------------------
print(20 * "-" + "ARMA (0,2): y(t) = e(t) + 0.5e(t-1) - 0.4e(t-2)" + 20 * "-")

T = int(input("Enter the number of data samples :"))
na = int(input("Enter AR order:"))
nb = int(input("Enter MA order:"))
mean = int(input("Enter Mean of white noise:"))
std = int(input("Enter variance of white noise:"))
scales = np.sqrt(std)
mean_y7 = (1*(1+0.5-0.4))/((1))

an = [0]*na
bn = [0]*nb

for i in range(na):
    an[i] = float(input("Enter coefficient of AR a{}".format(i+1)))

for i in range(nb):
    bn[i] = float(input("coefficient of MA b{}".format(i+1)))

max_order = max(na,nb)
num = [0]*(max_order+1)
den = [0]*(max_order+1)
for i in range(na+1):
    if i==0:
        den[i] = 1
    else:
        den[i] = an[i-1]

arparmas = np.array(an)
maparams = np.array(bn)

na = len(arparmas)
nb = len(maparams)

ar = np.r_[1, arparmas]
ma = np.r_[1, maparams]

arma_process = sm.tsa.ArmaProcess(ar,ma)
print("Is this stationary process: ", arma_process.isstationary)

y = arma_process.generate_sample(T, scales) + mean_y7

# ACF of the dependent variable.
autocorrelation = cal_auto_correlation(y, 15)
plot_acf(autocorrelation, "ACF plot for y")

#gpac
j = 7
k = 7
lags = j + k

# autocorrelation
ry = cal_auto_correlation(y, lags)
```

```
# create GPAC Table
gpac_table = create_gpac_table(j, k, ry)
print("GPAC Table for y:")
print(gpac_table.to_string())

# heatmap
plot_heatmap(gpac_table, "GPAC Table for y")

#
%%=================================================================================
============================
#8. ARMA (2,2): y(t)+0.5y(t-1) +0.2y(t-2) = e(t)+0.5e(t-1) - 0.4e(t-2)
# %%--------------------------------------------------------------------------------
------------------------------
print(20 * "-" + "ARMA (2,2): y(t)+0.5y(t-1) +0.2y(t-2) = e(t)+0.5e(t-1) - 0.4e(t-
2)" + 20 * "-")

T = int(input("Enter the number of data samples :"))
na = int(input("Enter AR order:"))
nb = int(input("Enter MA order:"))
mean = int(input("Enter Mean of white noise:"))
std = int(input("Enter variance of white noise:"))
scales = np.sqrt(std)
mean_y8 = (1*(1+0.5-0.4))/((1+0.5+0.2))

an = [0]*na
bn = [0]*nb

for i in range(na):
    an[i] = float(input("Enter coefficient of AR a{}".format(i+1)))

for i in range(nb):
    bn[i] = float(input("coefficient of MA b{}".format(i+1)))

max_order = max(na,nb)
num = [0]*(max_order+1)
den = [0]*(max_order+1)
for i in range(na+1):
    if i==0:
        den[i] = 1
    else:
        den[i] = an[i-1]

arparmas = np.array(an)
maparams = np.array(bn)

na = len(arparmas)
nb = len(maparams)

ar = np.r_[1, arparmas]
ma = np.r_[1, maparams]

arma_process = sm.tsa.ArmaProcess(ar,ma)
print("Is this stationary process: ", arma_process.isstationary)

y = arma_process.generate_sample(T, scales) + mean_y8
```

```python
# ACF of the dependent variable.
autocorrelation = cal_auto_correlation(y, 15)
plot_acf(autocorrelation, "ACF plot for y")

#gpac
j = 7
k = 7
lags = j + k

# autocorrelation
ry = cal_auto_correlation(y, lags)

# create GPAC Table
gpac_table = create_gpac_table(j, k, ry)
print("GPAC Table for y:")
print(gpac_table.to_string())

# heatmap
plot_heatmap(gpac_table, "GPAC Table for y")
```

REFERENCES

https://otexts.com/fpp2/#