Assignment 1

September 9,2022

Problem 1

The conditional expectation of the multivariate normal should be equal to the expected value from an OLS regression. Use the data in problem 1.csv to prove this empirically. Prove mathematically that the expected value from both equations are the same.

1.1 Prove it empirically

First, load the data.

```
data = pd.read_csv("problem1.csv")
x = data['x']
y = data['y']
data
```

	x	y
0	-1.166289	1.014680
1	-0.426878	0.262715
2	-1.477892	-1.044772
3	3.049119	0.804363
4	-2.123732	-0.689514
...	...	...
95	-0.588599	0.652704
96	-0.218138	0.067676
97	0.342822	1.214472
98	0.337376	0.608974
99	1.153817	-0.683444

100 rows × 2 columns

1.1 Prove it empirically

Then, calculate the conditional expectation of bivariate normal distribution.

```
y_bar = np.mean(y)
x_bar = np.mean(x)
cov_xy = np.cov([x,y])[0][1]
var_x = np.cov([x,y])[0][0]
mu = y_bar+cov_xy/var_x*(x-x_bar)
mu
```

```
0    -0.461299
1    -0.144828
2    -0.594666
3     1.342911
4    -0.871088
        ...
95   -0.214046
96   -0.055487
97    0.184606
98    0.182275
99    0.531715
Name: x, Length: 100, dtype: float64
```

1.1 Prove it empirically

Fit the data to the OLS regression model and calculate the fitted y values

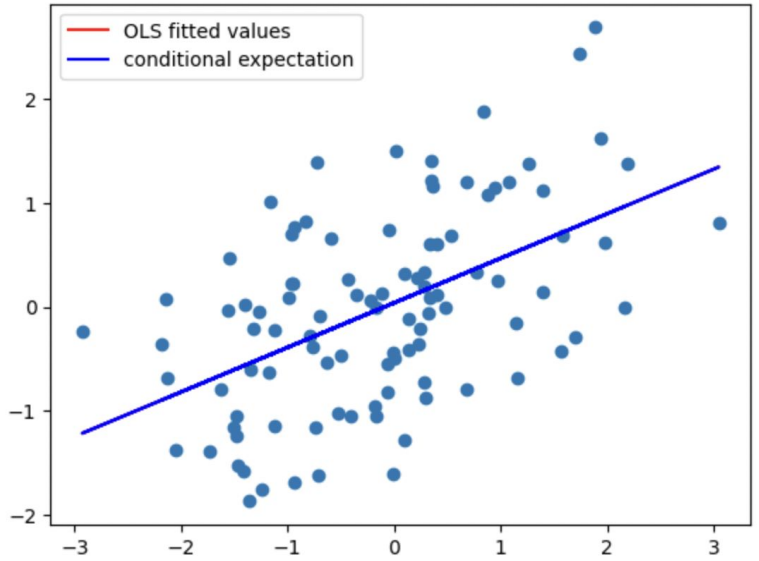
```
X = sm.add_constant(x)
model = sm.OLS(y,X)
results = model.fit()
fit_y = results.params['x']*x+results.params['const']
```

1.1 Prove it empirically

Plot those two values on a graph and calculate the difference between them

```
plt.plot(x,y,'o')
plt.plot(x,fit_y,color='r',label = 'OLS fitted values')
plt.plot(x,mu,color='b', label = 'conditional expectation')
plt.legend()
fit_y.tolist().sort()
mu.tolist().sort()
print(sum(fit_y-mu))
```

```
7.20430659573168e-15
```



1.1 Prove it empirically

Conclusion:From the graph, we can see that the line of fitted y(red line) and the line of the conditional expectation of bivariate normal overlap. We then calculate the sum of the difference between those sorted values and we got 7.2e^15 which is basically 0. Therefore, we proved that the conditional expectation of the multivariate normal is equal to the expected value from an OLS regression empirically.

1.2 Prove it mathematically

1.2 Prove it mathematically

From the lecture, we know that

$$\hat{\beta}_i = \frac{Cov(x_i, y)}{{\sigma_y}^2}$$

and we can the OLS for two variables is

$$y = \alpha + \beta x + \epsilon$$

Since the error term disappears after taking the expectation on both sides, what we really have is

$$\hat{y} = \hat{\alpha} + \hat{\beta} x$$

We also know that for OLS

$$\hat{\alpha} = \bar{y} - \hat{\beta}\bar{x}$$

Combining the two equations above, we get

$$\hat{y} = \bar{y} - \hat{\beta}\bar{x} + \hat{\beta} x$$

which is

$$\hat{y} = \bar{y} + \hat{\beta}(x - \bar{x})$$

From the lecture, we know that

$$\beta = (X'X)^{-1}X'Y$$

and X in this case is [1 x] and β in this case is [α β]. Therefore, we can find that β in our problem is

$$\frac{Cov(x, y)}{Var(x)}$$

Therefore, we have the expectation for $\mu = \bar{y} + \frac{Cov(x,y)}{Var(x)}(x - \bar{x})$ and the expectation for fitted OLS$\hat{y} = \bar{y} + \frac{Cov(x,y)}{Var(x)}(x - \bar{x})$ and they are exactly the same.

Problem 2

Fit the data in problem2.csv using OLS and calculate the error vector. Look at its distribution. How well does it fit the assumption of normally distributed errors?
Fit the data using MLE given the assumption of normality. Then fit the MLE using the assumption of a T distribution of the errors. Which is the best fit?
What are the fitted parameters of each and how do they compare? What does this tell us about the breaking of the normality assumption in regards to expected values in this case?

Problem 2

First, load the data and fit the OLS model

```
data = pd.read_csv("problem2.csv")
x = data['x']
y = data['y']
X = sm.add_constant(x)
model = sm.OLS(y,X)
results = model.fit()
results.params
```

```
const    0.119836
x        0.605205
dtype: float64
```

Problem 2

Calculate the error vector base on the model and fitted values

```
error = y - results.params['x']*x+results.params['const']
error
```

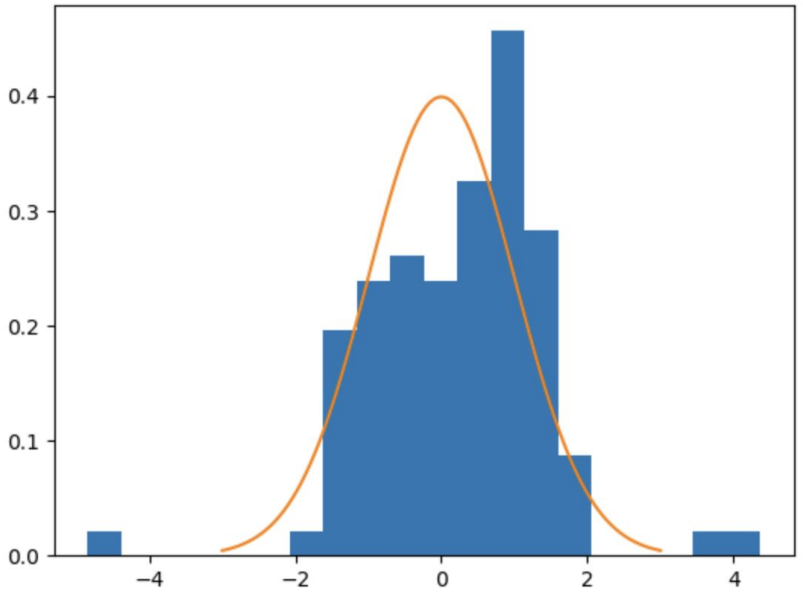
```
0    -0.598812
1     1.074968
2     1.267101
3     1.559383
4     0.087356
        ...
95   -1.350591
96   -1.455176
97    0.674550
98    0.641934
99   -0.682646
Length: 100, dtype: float64
```

Problem 2

Draw a histogram for the distribution of the error vector and we found that it does not fit

```
plt.hist(error, bins = 20, density = True)
x_axis = np.linspace(-3, 3, 100)
plt.plot(x_axis, norm.pdf(x_axis, 0,1))
```

```
[<matplotlib.lines.Line2D at 0x2a29cee00>]
```



Problem 2

Fit the model using MLE for two assumptions on the error vectors:normal distribution and T distribution. Get the parameters of the two models.

```
from scipy import stats
from scipy.optimize import minimize

def MLE_N(p):
    fitted = p[0] + p[1]*x
    return -1*np.sum(stats.norm.logpdf(y, fitted, p[2]))
def MLE_T(p):
    fitted = p[0] + p[1]*x
    return -1*np.sum(stats.t.logpdf(y-fitted, p[2],scale=p[3]))
mle_N = minimize(MLE_N, x0=(1,1,1))
mle_T = minimize(MLE_T, x0=(1,1,10,1))
print(mle_N.x,mle_T.x)
```

```
[0.11983616 0.60520482 1.19839405] [0.14261403 0.55757181 6.27655964 0.97126593]
```

Problem 2

Use R-Square test to see which model is better and it turns out to be normality assumption model

```
SS_total = sum([i**2 for i in (y-np.mean(y))])
error_N = y - 0.60520482*x+0.11983616
SS_error_N = sum([i**2 for i in (error_N-np.mean(error_N))])
RR_N = 1-SS_error_N/SS_total
error_T = y - 0.55757181*x+0.14261403
SS_error_T = sum([i**2 for i in (error_T-np.mean(error_T))])
RR_T = 1-SS_error_T/SS_total
RR_N,RR_T
```

```
(0.194639523918951, 0.19343381555614025)
```

Problem 2

Use AIC and BIC criteria for two models to see which one is better. Still normality assumption model.

```
AIC_N = 2*3+2*mle_N.fun
AIC_T = 2*4+2*mle_T.fun
AIC_N,AIC_T
```

```
(325.98419337832564, 318.9459408249327)
```

```
BIC_N = 3*np.log(len(y))+2*mle_N.fun
BIC_T = 4*np.log(len(y))+2*mle_T.fun
BIC_N,BIC_T
```

```
(333.7997039362899, 329.366621568885)
```

Problem 2

Conclusion

Therefore, we conclude that the model that assumes error has a normal distribution is better because it shows a better fit in all 3 criterias. The parameters for normal distribution assumption model are $\alpha=0.11983616$ and $\beta=0.60520482$.The parameters for T distribution assumption model are $\alpha=0.14261403$ and $\beta=0.55757181$. This tells us that with different assumptions on the distribution of the error vector, the fitted values will be different and therefore, it's important to find which distribution the error vector fits into before we do the OLS regression. We can't not break the normality assumption in this case as the normality assumption model is better than T-distribution assumption model.

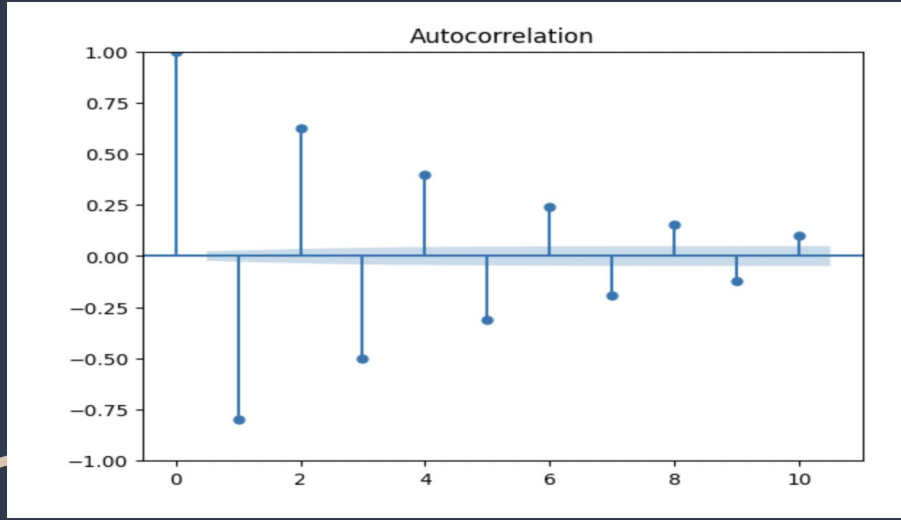
Problem 3

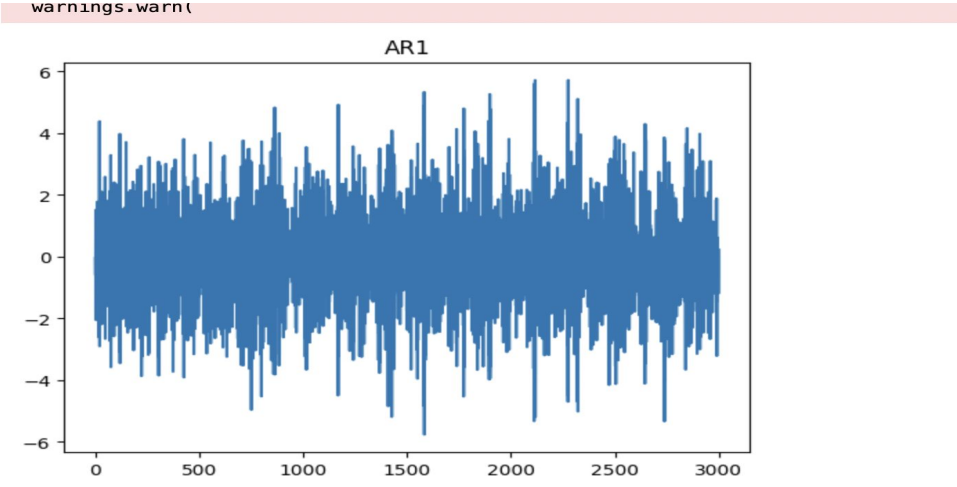
Code

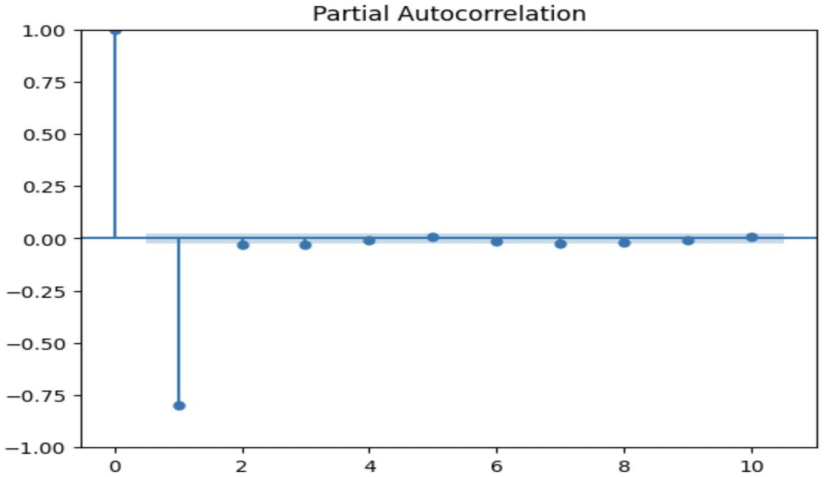
```
from statsmodels.tsa.arima_process import ArmaProcess
from statsmodels.graphics.tsaplots import plot_pacf,plot_acf
AR1_process = ArmaProcess(ar=[1,0.8])
simulation = AR1_process.generate_sample(nsample=3000)
plt.plot(simulation)
plt.title("AR1")
fig = plot_acf(simulation, alpha=0.2, lags=10)
fig = plot_pacf(simulation, alpha=0.2, lags=10)
```

Problem 3

AR(1)

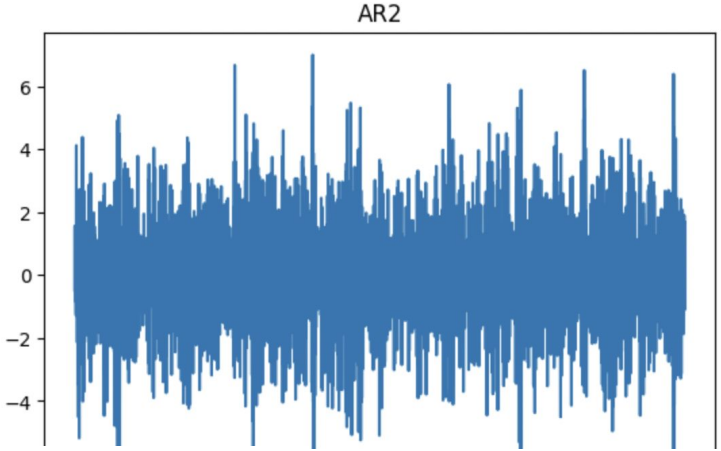


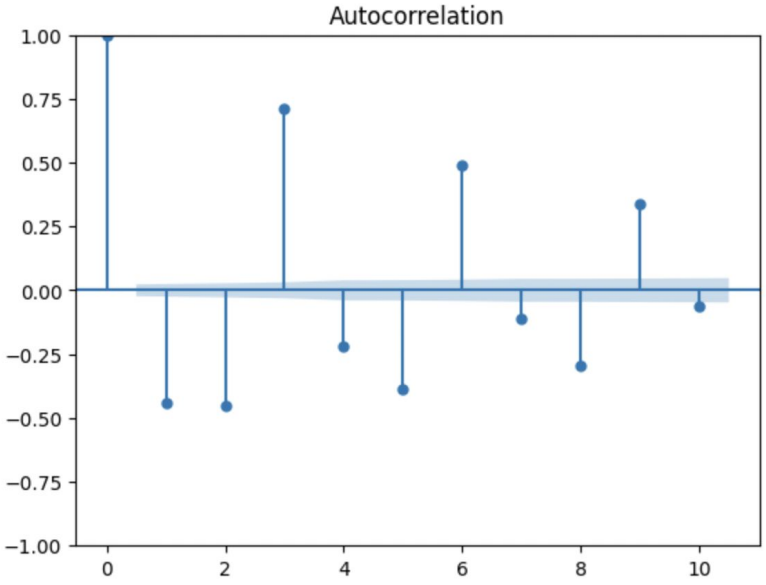


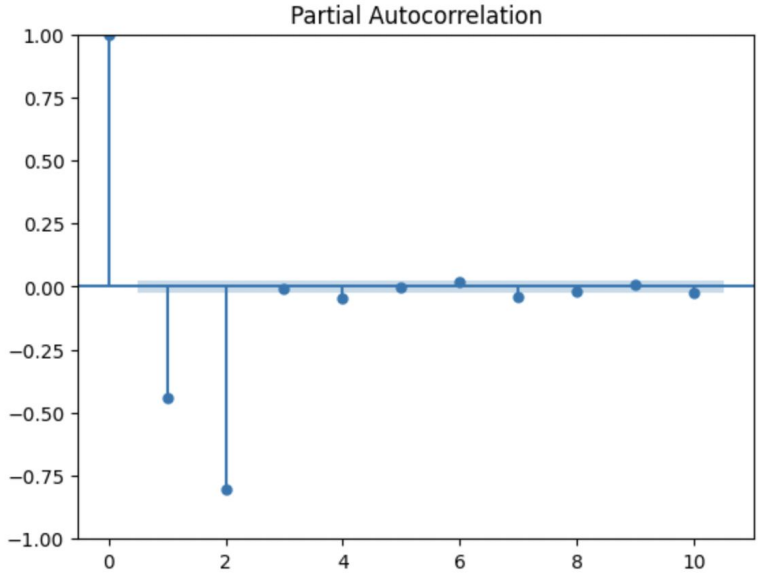


Problem 3

AR(2)

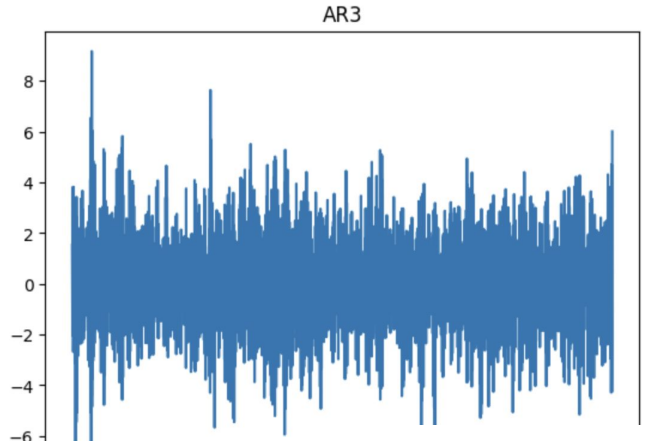


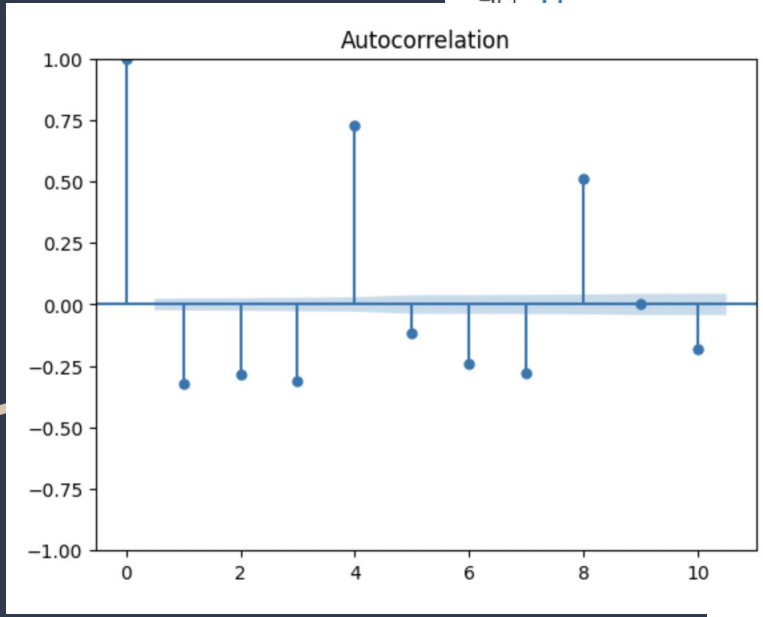


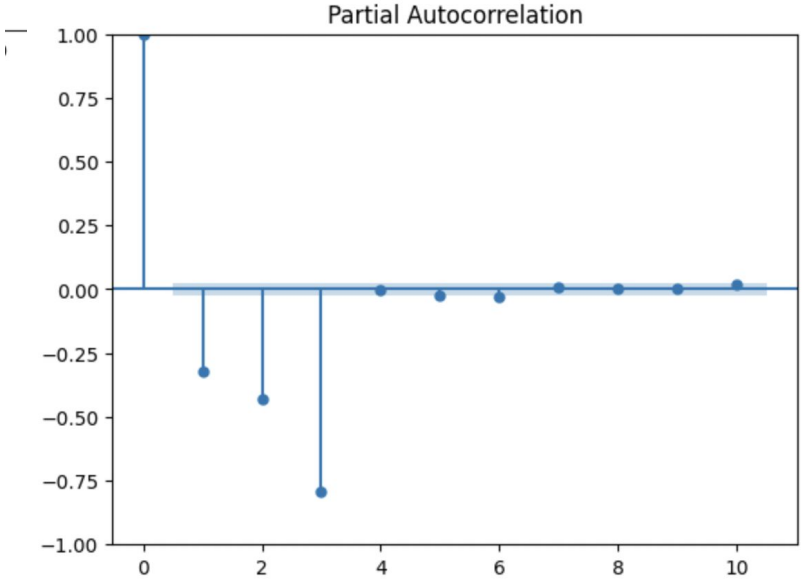


Problem 3

AR(3)

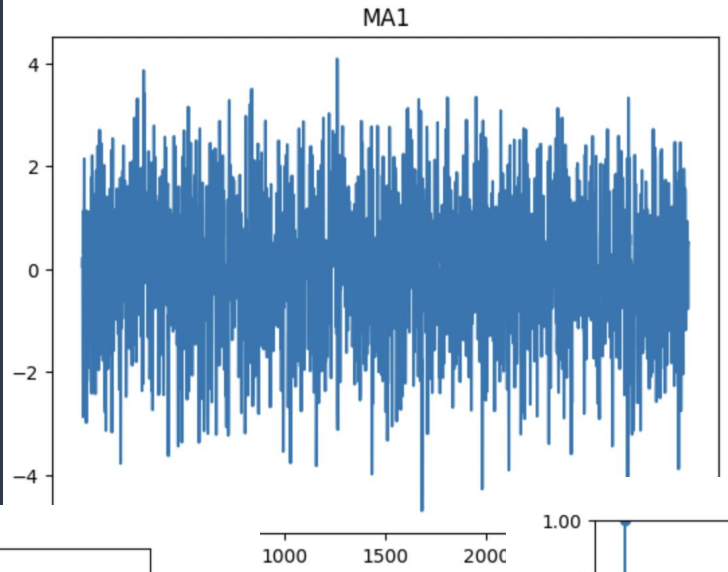


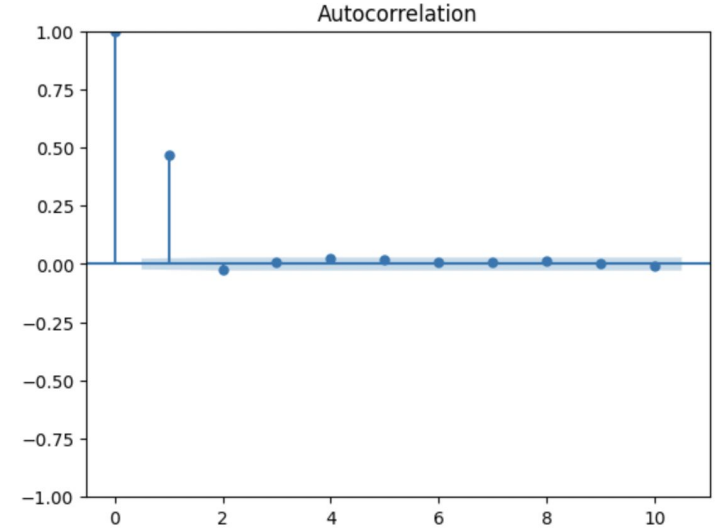


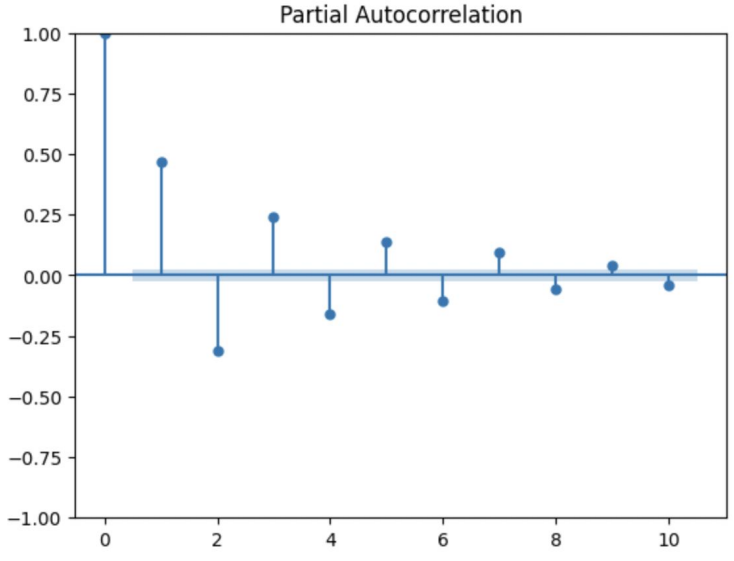


Problem 3

MA(1)

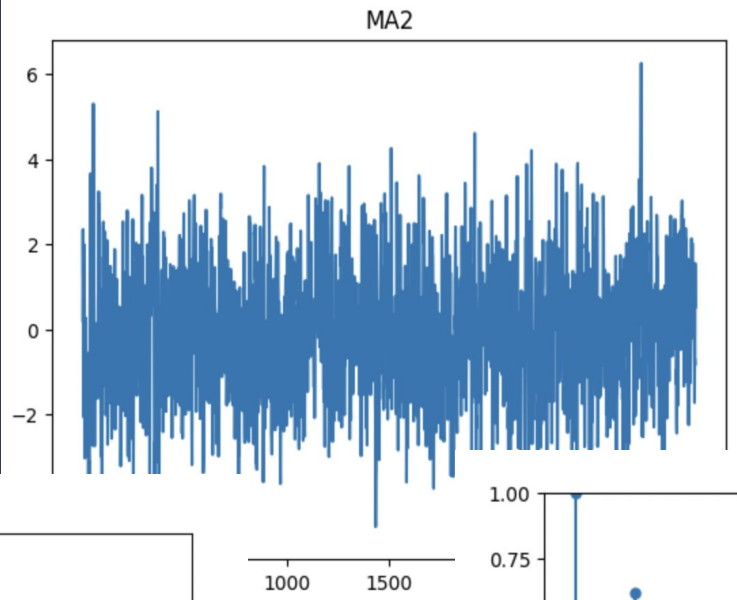


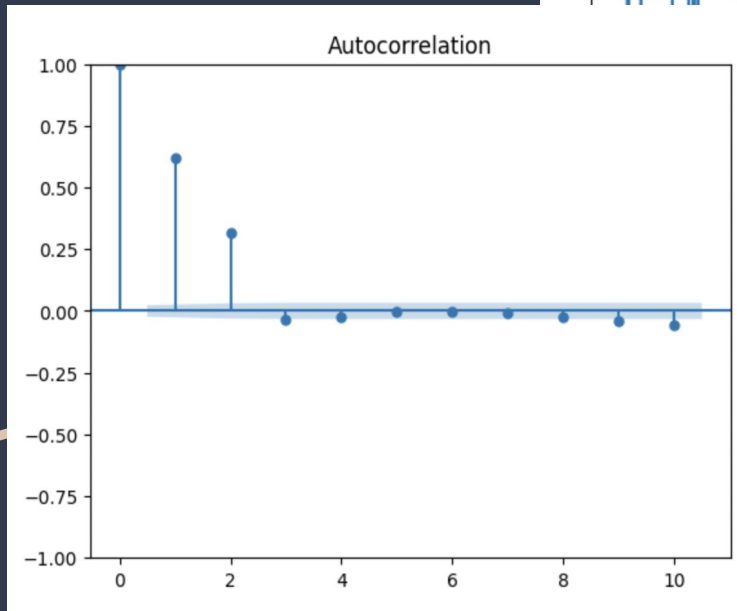


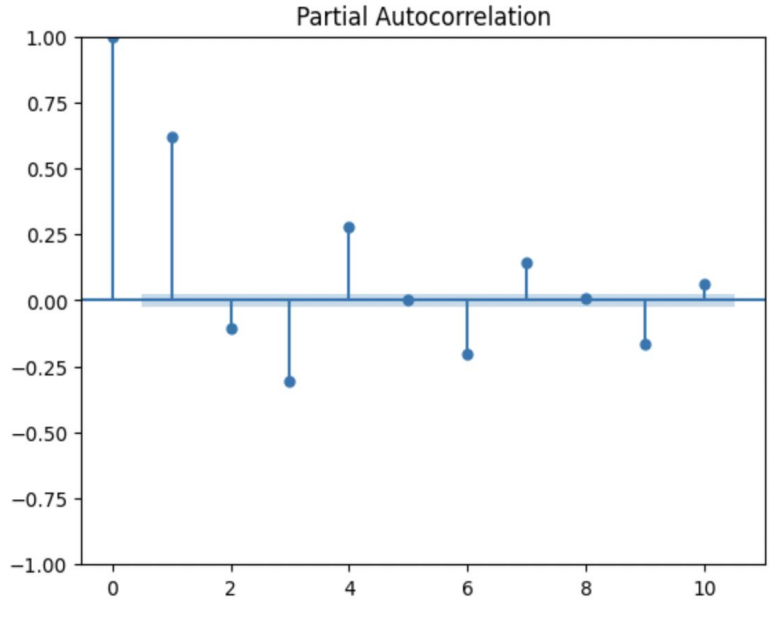


Problem 3

MA(2)

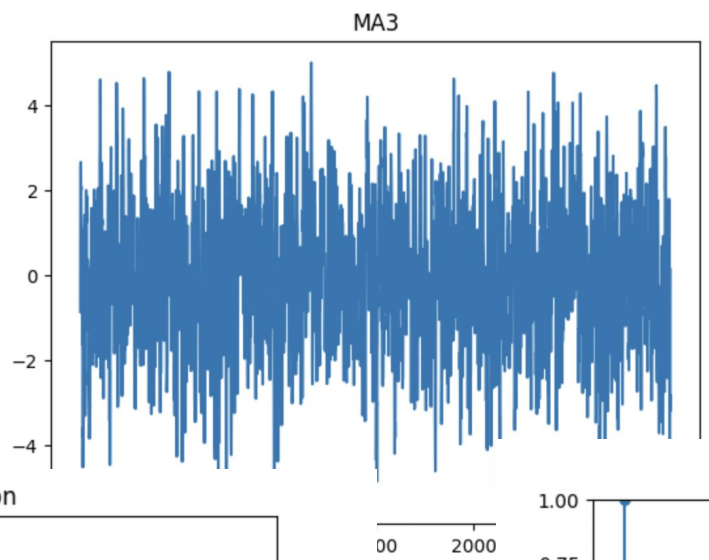


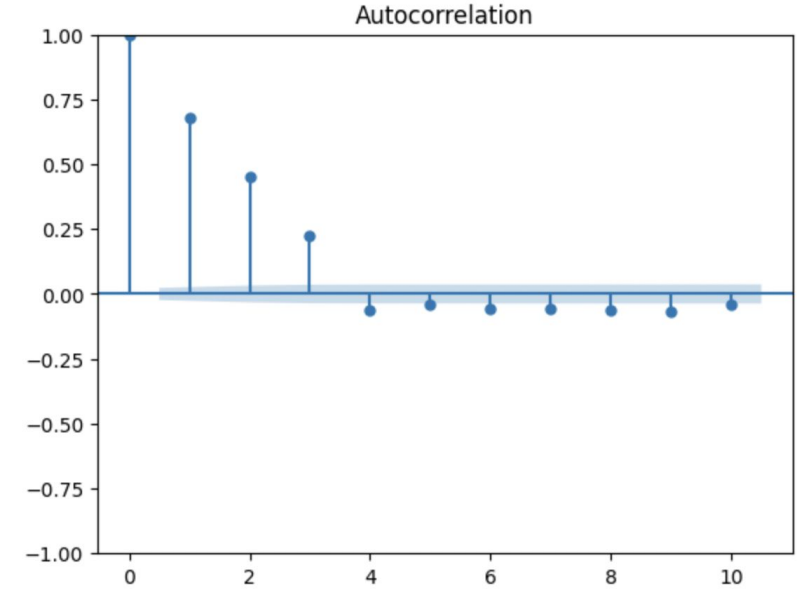


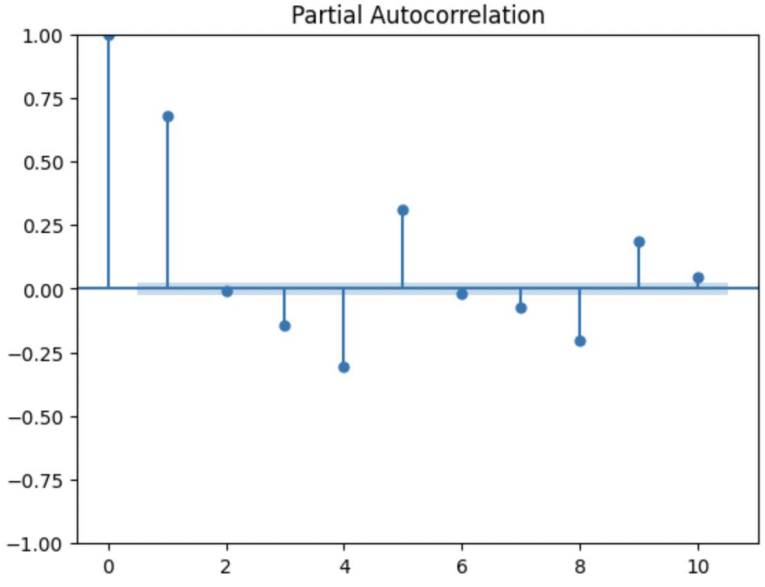


Problem 3

MA(3)







Problem 3

How to determine the type and order of a process from ACF and PCF?

First we look at the graphs of ACF. If the absolute value of lags drops immediately to 0 at some point, then the process is MA and the order is the number of lags with non-zero values -1. If the absolute value of lags decreases in a smooth trend, then the process is AR and the order of the process is the number of lags with negative values consecutively. Next, we look at the graphs of PCF. If the value of lags immediately to 0 at some point, then the process is AR and the order is the number of lags with negative values. If the value of lags fluctuates, then the process is MA and the order is the number of lags between two ridges.