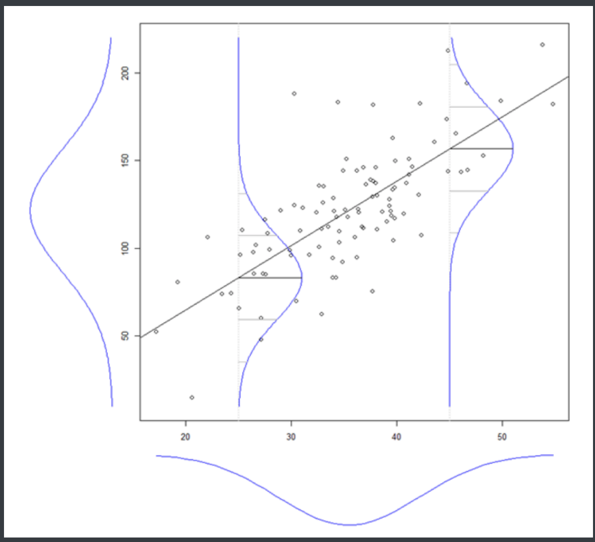
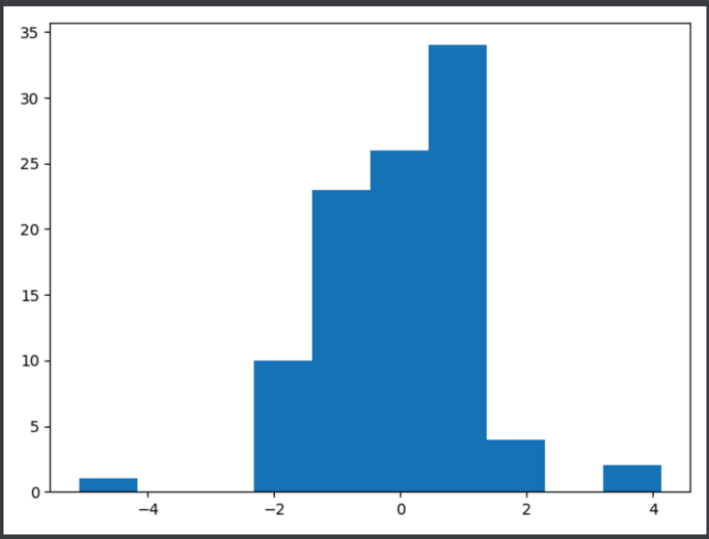
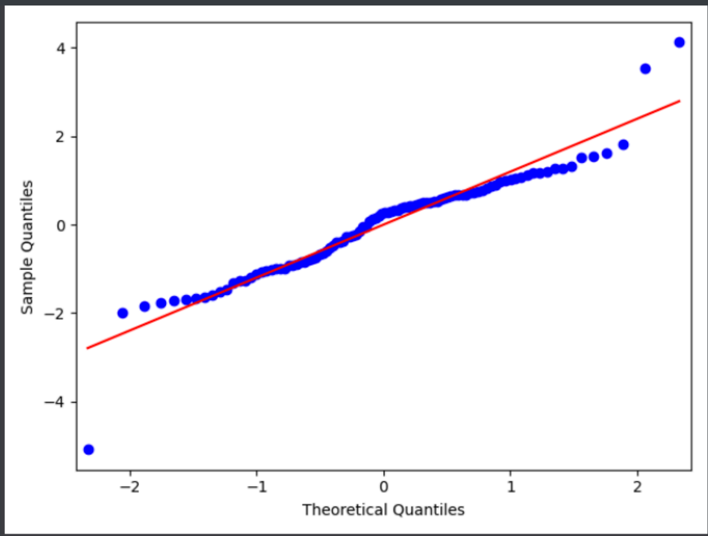
Project Response

Problem 1

The conditional distribution of the Multivariate Normal to the OLS equations are the same.

- ```
      conExpY   fittedY
  0  -0.461299 -0.461299
  1  -0.144828 -0.144828
  2  -0.594666 -0.594666
  3   1.342911  1.342911
  4  -0.871088 -0.871088
  ..       ...       ...
  95 -0.214046 -0.214046
  96 -0.055487 -0.055487
  97  0.184606  0.184606
  98  0.182275  0.182275
  99  0.531715  0.531715

  [100 rows x 2 columns]
  ```

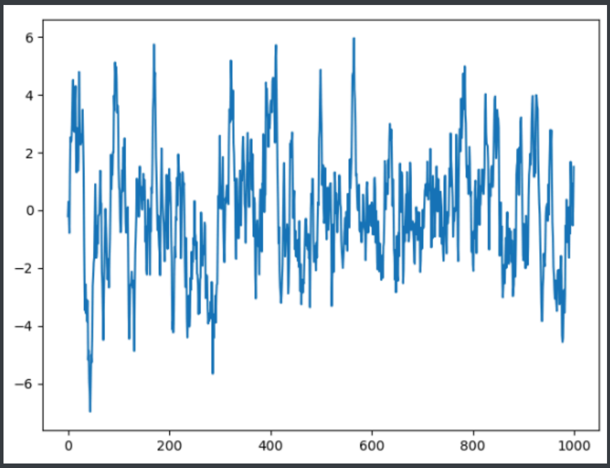
Reason

- OLS is to estimate Y given X, which has the same meaning of finding the conditional distribution of Y given X. If the covariance between X and Y is not 0, the mean of the conditional distribution of Y has to adjust according to the covariance between X and Y.
  ```

- 

- This graph vividly shows the reason why the two are the same. The graph shows that the margin distribution of X and Y are both normal. Also, X and Y are positively correlated. the fitted Y given X=x0 is exactly the mean of Y's normal distribution given X=x0. The conditional mean adjusts when X changes.

Problem 2

OLS

Distribution of the error vector using OLS



How well does it fit the assumption of normally distributed errors?

- To know how well it fits the assumption of normally distributed errors, I use qqplot and SW test to estimate its normality. qqplot shows the distribution of the data against the expected normal distribution. For normally distributed data, observations should lie approximately on a straight line. Shapiro-Wilk test is a test of normality.
    - qqplot



    - SW test

```
Statistics=0.938, p=0.000
Sample does not look Gaussian (reject H0)
```

- As seen in the pictures, the histogram of the error vector is skewed, the qqplot is not straight, and the SW test reject the hypothesis that the Sample is an Gaussian.
- Therefore, it does not fit the assumption of normally distributed errors.

MLE

MLE with normal assumption and t assumption

```
R square for MLE with normal distributed errors:  0.1946395239189488
R square for MLE with t distributed errors:  0.19457897480090924
```

- The MLE using the assumption of a T distribution of the errors is the best fit.

Comparing parameters

What are the fitted parameters of each and how do they compare?

- The fitted parameters of OLS are the same as the fitted parameters of MLE with normal assumption.
- The fitted parameters of MLE with normal assumption are different from those of the MLE with t assumption.
- ```
  OLS:
  Intercept    0.119836
  x            0.605205
  dtype: float64

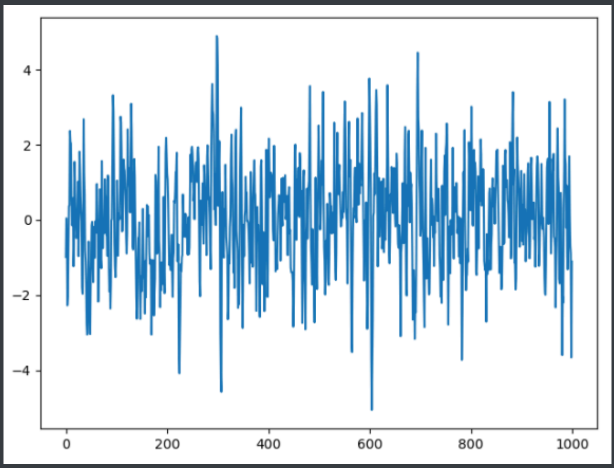

  MLE with normal assumption:
  fitted intercept:  0.11983623040349767
  fitted beta:  0.6052048639645156


  MLE with t assumption:
  fitted intercept:  0.12325306418741294
  fitted beta:  0.5951244744180728
  ```

What does this tell us about the breaking of the normality assumption in regards to expected values in this case?

- The breaking of the normality assumption will influence parameter estimation

Problem 3

Simulate AR(1) through AR(3) and MA(1) through MA(3) processes
  ```

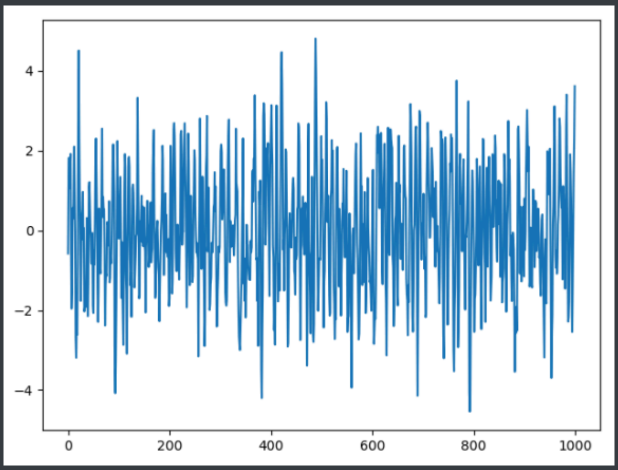
```
# AR1
ar = np.array([1, -0.9])
ma = np.array([1])
AR = ArmaProcess(ar, ma)
simulated_ar1 = AR.generate_sample(nsample=1000)
plt.plot(simulated_ar1)
plt.show()
```



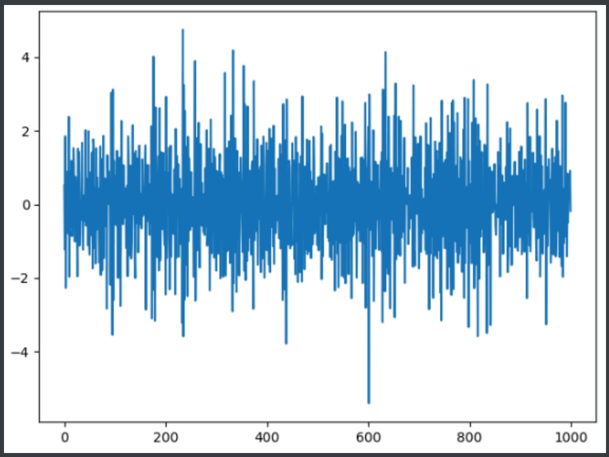

```
# AR2
ar = np.array([1, -0.9, 0.3])
ma = np.array([1])
AR = ArmaProcess(ar, ma)
simulated_ar2 = AR.generate_sample(nsample=1000)
plt.plot(simulated_ar2)
plt.show()
```



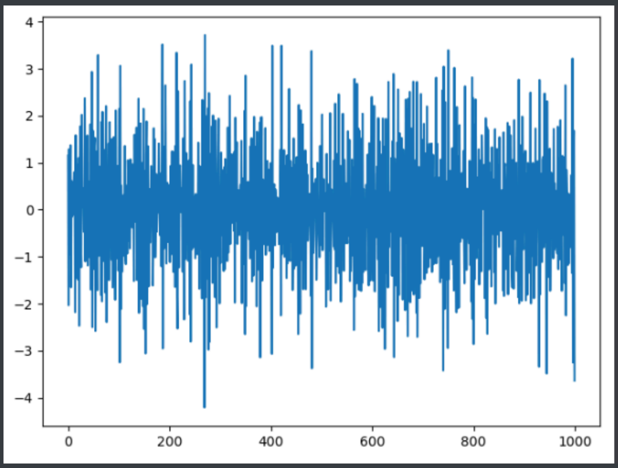
```python
# AR3
ar = np.array([1, -0.9, 0.3, 0.2])
ma = np.array([1])
AR = ArmaProcess(ar, ma)
simulated_ar3 = AR.generate_sample(nsample=1000)
plt.plot(simulated_ar3)
plt.show()
```



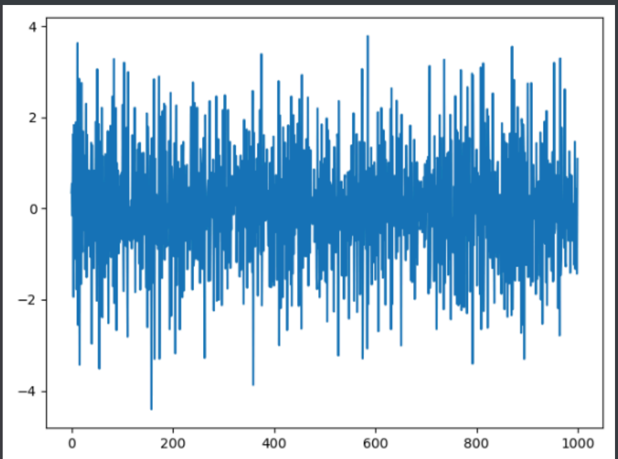

```python
# MA1
ar1 = np.array([1])
ma1 = np.array([1, -0.9])
MA = ArmaProcess(ar1, ma1)
simulated_ma1 = MA.generate_sample(nsample=1000)
plt.plot(simulated_ma1)
plt.show()
```



```
# MA2
ma = np.array([1, -0.9, 0.3])
ar = np.array([1])
MA = ArmaProcess(ar, ma)
simulated_ma2 = MA.generate_sample(nsample=1000)
plt.plot(simulated_ma2)
plt.show()
```



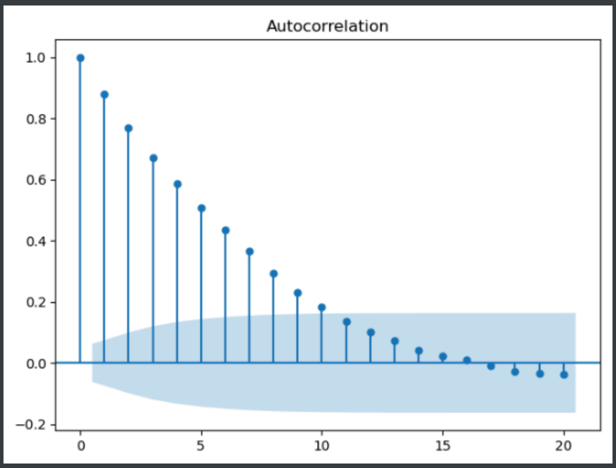

```
# MA3
ma = np.array([1, -0.9, 0.3, 0.2])
ar = np.array([1])
MA = ArmaProcess(ar, ma)
simulated_ma3 = MA.generate_sample(nsample=1000)
plt.plot(simulated_ma3)
plt.show()
```



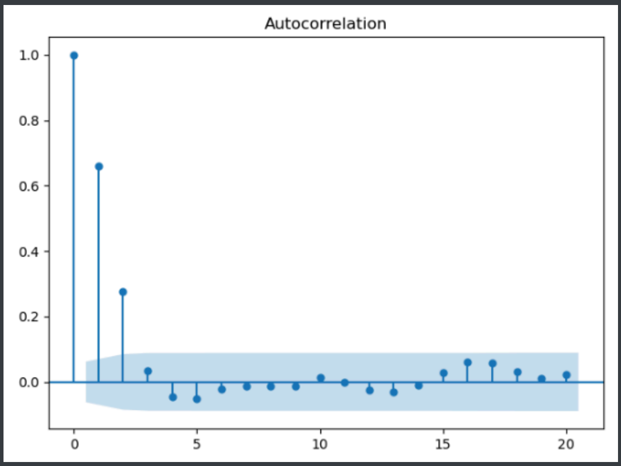
ACF and PACF

ACF

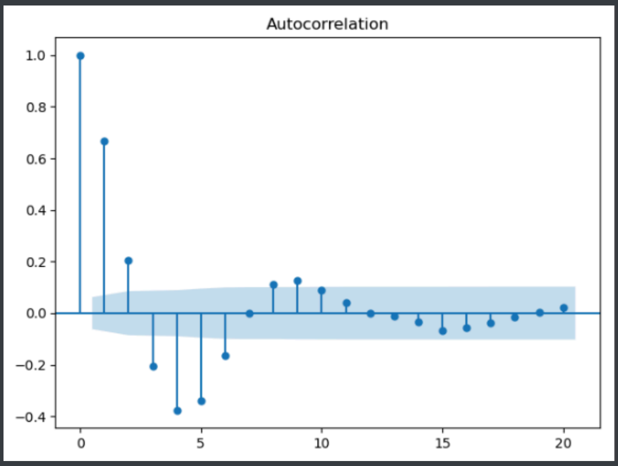
- AR1



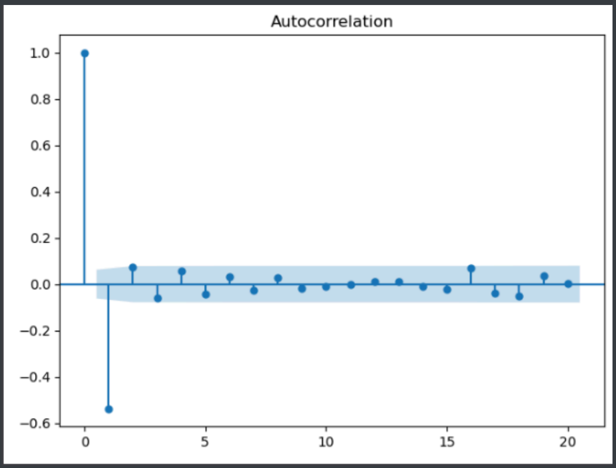
- AR2



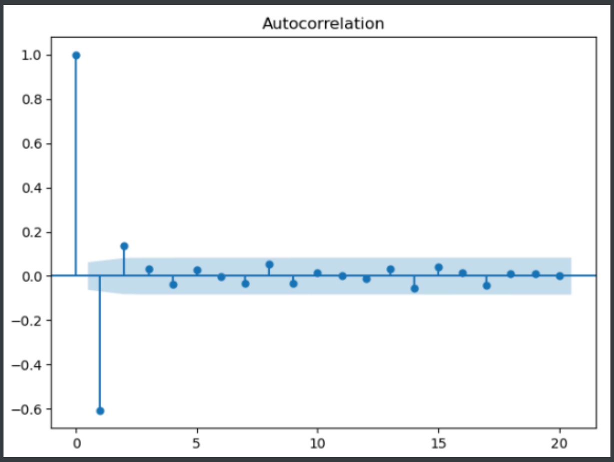
- AR3



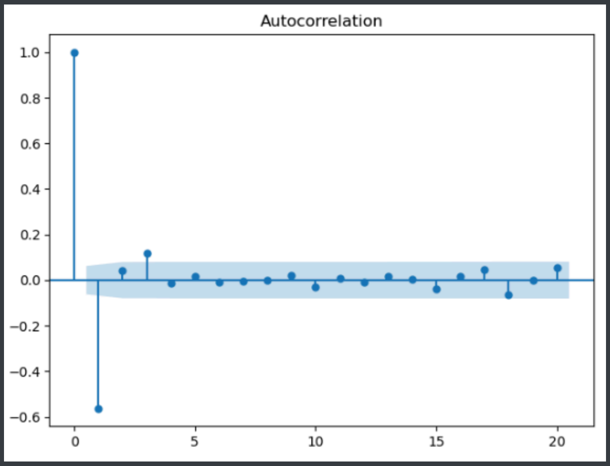
- MA1



- MA2

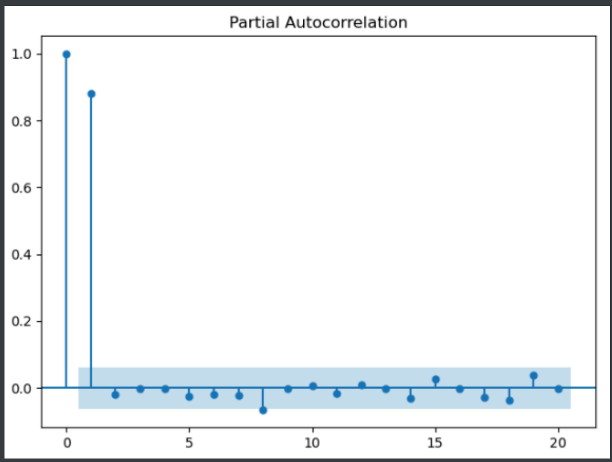


- MA3

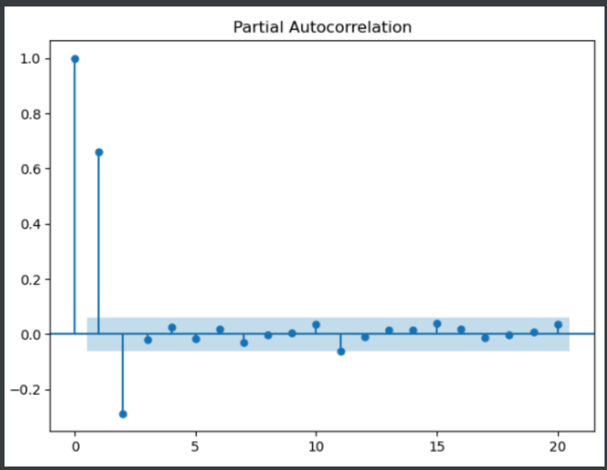


PACF

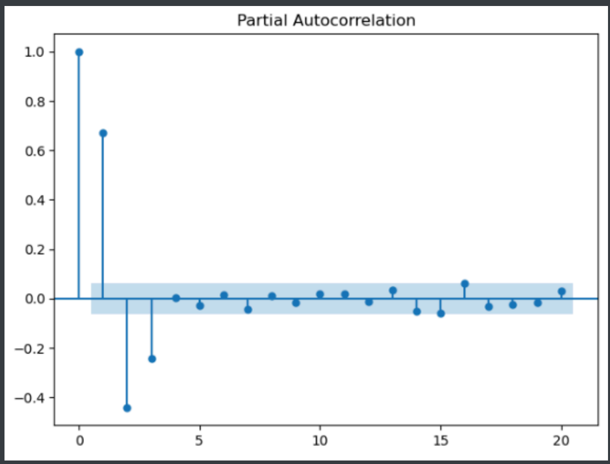
- AR1


Partial Autocorrelation
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20

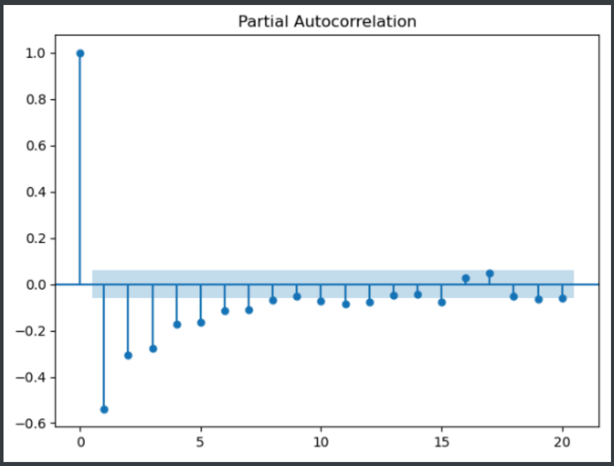
- AR2


Partial Autocorrelation
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0
5
10
15
20

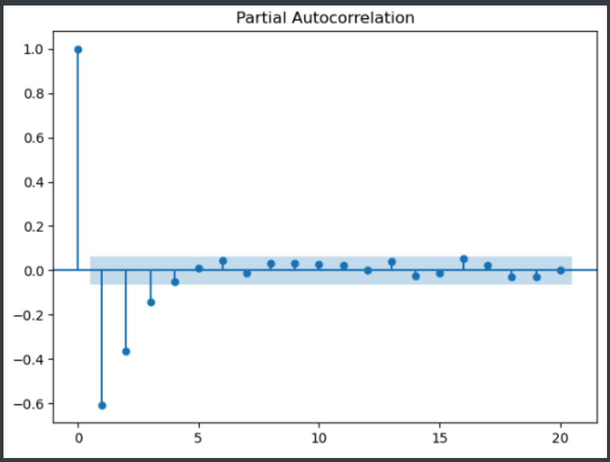
- AR3


Partial Autocorrelation
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
0
5
10
15
20

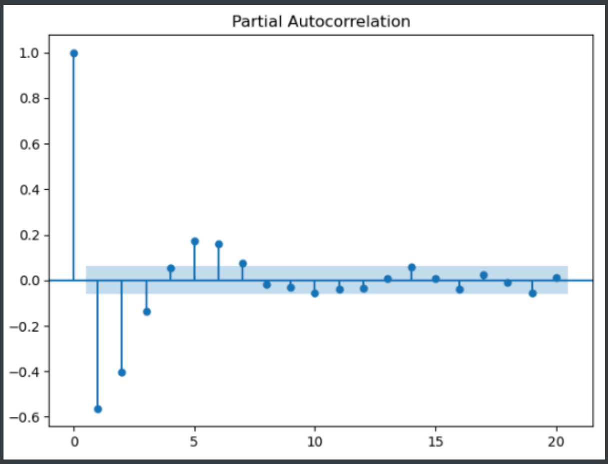
- MA1



- MA2



- MA3



Identify the type and order of each process

- If a process is an AR process, its autocorrelation will decrease, or oscillate to decrease slowly, and the number of lags that are significantly differ from 0 in the partial autocorrelation indicates the order of this AR process.

- If a process is an MA process, its partial autocorrelation will decrease, or oscillate to decrease slowly, and the number of lags that are significantly differ from 0 in the autocorrelation indicates the order of this MA process.