Importing required libraries

In [28]:
```
import pandas as pd
from statsmodels.tsa.arima.model import ARIMA
import matplotlib.pyplot as plt
import warnings
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
import statsmodels.api as sm
from sklearn.metrics import mean_squared_error

warnings.filterwarnings("ignore")

plt.rcParams["figure.figsize"] = (10,5)
```

Question 1

The data provided in the file Measurement_Q1.xls exhibits a linear trend. Apply the following models to the data. (20 pts)

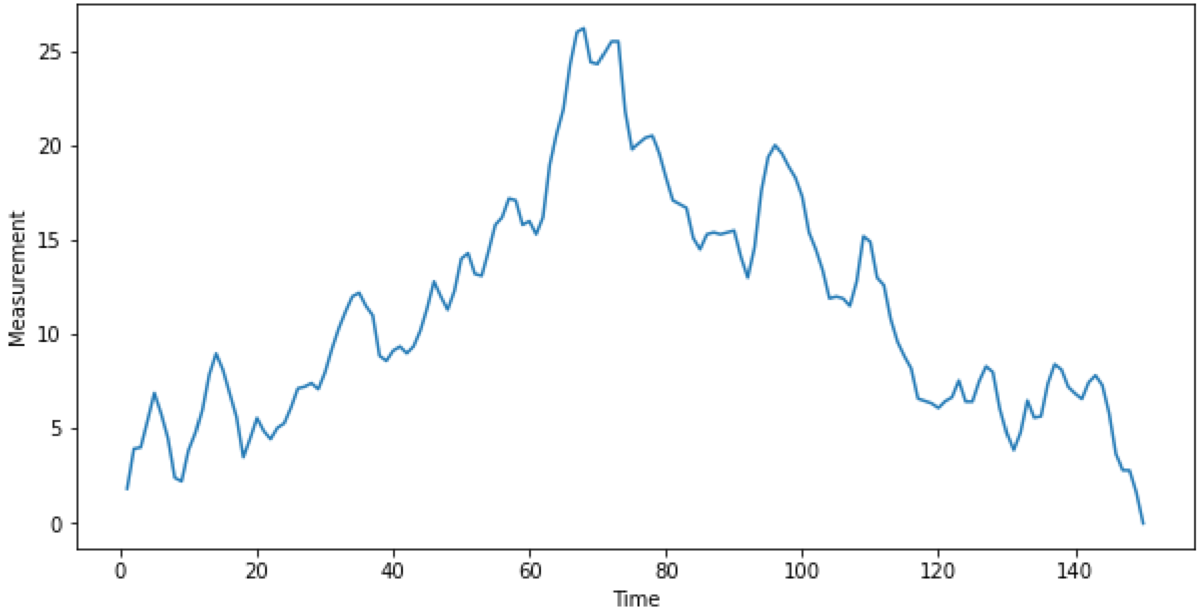
In [8]:
```
df = pd.read_csv('Measurement_Q1.csv', index_col = 'Time')
df
```

Out[8]:

Time	Measurement
1	1.84
2	3.93
3	4.00
4	5.42
5	6.89
...	...
146	3.65
147	2.82
148	2.81
149	1.64
150	0.00

150 rows × 1 columns

In [9]:
```
plt.xlabel('Time')
plt.ylabel('Measurement')
plt.plot(df)
plt.show()
```



Part a

Develop an IMA(1,1) model for the data. Display the model parameters obtained as your output (8 pts)

In [10]:
```
model = ARIMA(df.Measurement, order=(0,1,1))
model_fit = model.fit()
model_fit.summary()
```

Out[10]:

SARIMAX Results

Dep. Variable:	Measurement	**No. Observations:**	150
Model:	ARIMA(0, 1, 1)	**Log Likelihood**	-202.609
Date:	Wed, 12 Oct 2022	**AIC**	409.217
Time:	12:36:16	**BIC**	415.225
Sample:	0	**HQIC**	411.658
	- 150		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ma.L1	0.7531	0.060	12.573	0.000	0.636	0.871
sigma2	0.8834	0.109	8.080	0.000	0.669	1.098

Ljung-Box (L1) (Q):	0.26	**Jarque-Bera (JB):**	1.10
Prob(Q):	0.61	**Prob(JB):**	0.58
Heteroskedasticity (H):	1.00	**Skew:**	-0.21
Prob(H) (two-sided):	0.99	**Kurtosis:**	2.98

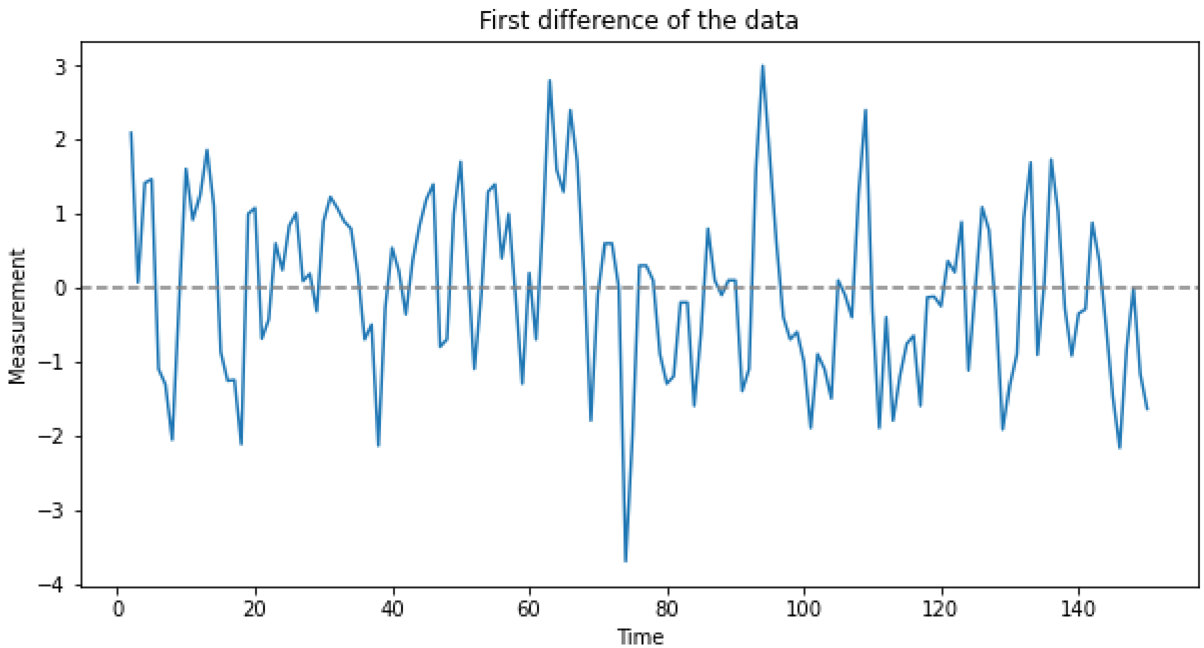
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

Part b

Compute and plot the first difference of the data. (2 pts)

In [12]:
```
df_diff1 = sm.tsa.statespace.tools.diff(df.Measurement, k_diff=1)
plt.plot (df_diff1)
plt.axhline(y=0, color='grey', linestyle='--')
plt.xlabel ('Time')
plt.ylabel ('Measurement')
```

```
plt.title ('First difference of the data')
plt.show()
```



Part c

Now, develop an MA(1) model on the first difference. Display the model parameters obtained as your output (8 pts)

In [15]:
```
model = ARIMA(df_diff1, order=(0,0,1))
model_fit = model.fit()
model_fit.summary()
```

Out[15]:

SARIMAX Results

Dep. Variable:	Measurement	**No. Observations:**	149
Model:	ARIMA(0, 0, 1)	**Log Likelihood**	-202.607
Date:	Wed, 12 Oct 2022	**AIC**	411.215
Time:	12:38:58	**BIC**	420.227
Sample:	0	**HQIC**	414.876
	- 149		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
const	-0.0064	0.137	-0.047	0.962	-0.274	0.261
ma.L1	0.7531	0.061	12.392	0.000	0.634	0.872
sigma2	0.8834	0.110	8.052	0.000	0.668	1.098

Ljung-Box (L1) (Q):	0.26	**Jarque-Bera (JB):**	1.10
Prob(Q):	0.61	**Prob(JB):**	0.58
Heteroskedasticity (H):	0.99	**Skew:**	-0.21
Prob(H) (two-sided):	0.98	**Kurtosis:**	2.98

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

Part d

Based on the model parameters you obtained in (a) and (c), comment on how the two models are related. (2 pts)

The constant and theta_1 are same for IMA(1,1) model and MA(1) model of first difference:

constant = -0.0064 Theta_1 = 0.7531

This is because IMA(1,1) has d = 1 and q = 1. This just means that the first difference is calculated automatically and it includes a 1 MA term.

for the second model the differencig is performed manually hence d = 1 and there is also n MA 1 term included so d = 1 amd q = 1 even with the second model.

Question 2

Consider the global mean surface air temperature anomaly data provided in GlobalAirTemperature.xls. (15 pts)

In [16]:
```
df = pd.read_csv('GlobalAirTemperature.csv', index_col = 'Year')
df
```

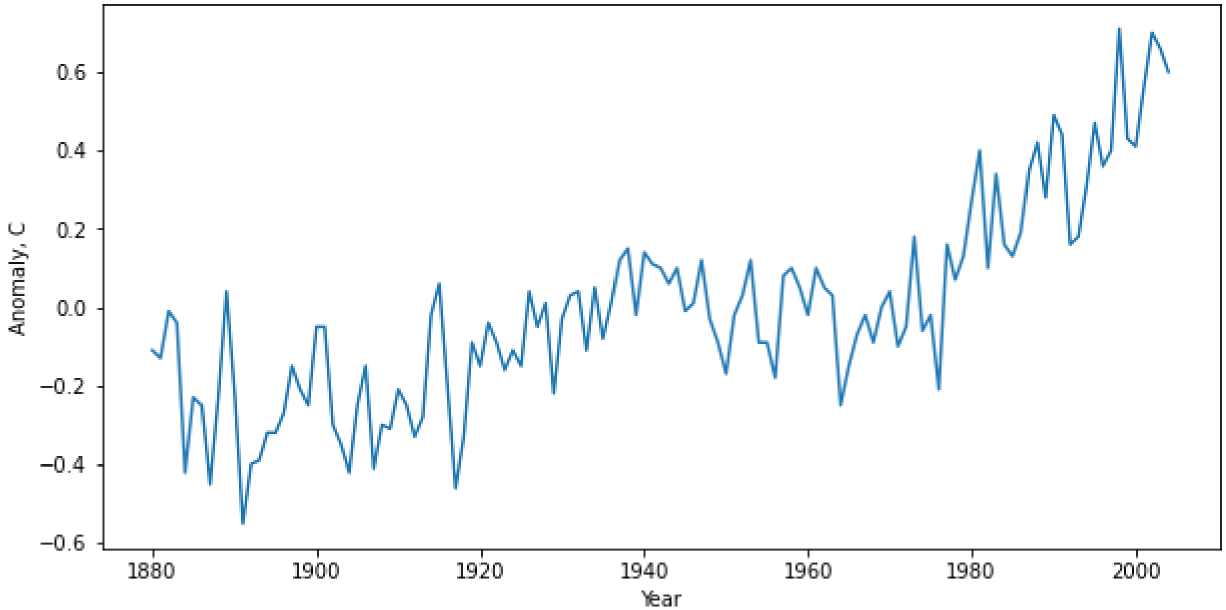
Out[16]:

Year	Anomaly, C
1880	-0.11
1881	-0.13
1882	-0.01
1883	-0.04
1884	-0.42
...	...
2000	0.41
2001	0.56
2002	0.70
2003	0.66
2004	0.60

125 rows × 1 columns

In [17]:
```
plt.xlabel('Year')
plt.ylabel('Anomaly, C')
```

```
plt.plot(df)
plt.show()
```



Part a

Apply an IMA(1,1) model to this data. Calculate the SSE by comparing the model output with the data. (6 pts)

In [18]:
```
model = ARIMA(df['Anomaly, C'], order=(0,1,1))
model_fit = model.fit()
model_fit.summary()
```

Out[18]:

SARIMAX Results

Dep. Variable:	Anomaly, C	**No. Observations:**	125
Model:	ARIMA(0, 1, 1)	**Log Likelihood**	72.910
Date:	Wed, 12 Oct 2022	**AIC**	-141.821
Time:	12:51:14	**BIC**	-136.180
Sample:	0	**HQIC**	-139.530
	- 125		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ma.L1	-0.6640	0.072	-9.243	0.000	-0.805	-0.523
sigma2	0.0180	0.002	8.182	0.000	0.014	0.022

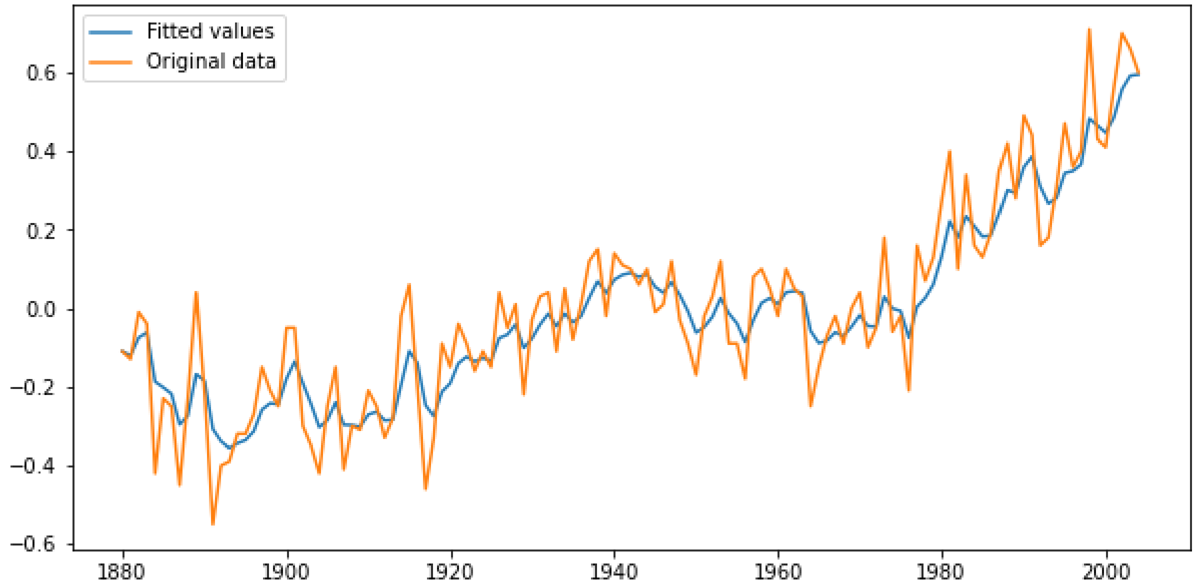
Ljung-Box (L1) (Q):	2.36	**Jarque-Bera (JB):**	2.20
Prob(Q):	0.12	**Prob(JB):**	0.33
Heteroskedasticity (H):	0.87	**Skew:**	-0.29
Prob(H) (two-sided):	0.65	**Kurtosis:**	3.32

Warnings:

[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [34]:

```
df_pred = model_fit.predict(start=1,end=len(df['Anomaly, C']), typ = 'levels').rename("Predictions")
plt.plot (df.index, df_pred, label='Fitted values')
plt.plot (df.index, df['Anomaly, C'], label='Original data')
plt.legend()
plt.show()
```



In [32]:
```
sse = mean_squared_error (df['Anomaly, C'].values, df_pred.values) * len(df_pred)
sse,len(df_pred.values)
```

Out[32]: (0.9811083887548632, 125)

Part b

Now, apply an IMA(1,2) model. Calculate the SSE. (6 pts)

In [36]:
```
model = ARIMA(df['Anomaly, C'], order=(0,1,2))
model_fit = model.fit()
model_fit.summary()
```

Out[36]:

SARIMAX Results

Dep. Variable:	Anomaly, C	**No. Observations:**	125
Model:	ARIMA(0, 1, 2)	**Log Likelihood**	76.508
Date:	Wed, 12 Oct 2022	**AIC**	-147.016
Time:	12:58:43	**BIC**	-138.555
Sample:	0	**HQIC**	-143.579
	- 125		
Covariance Type:	opg		

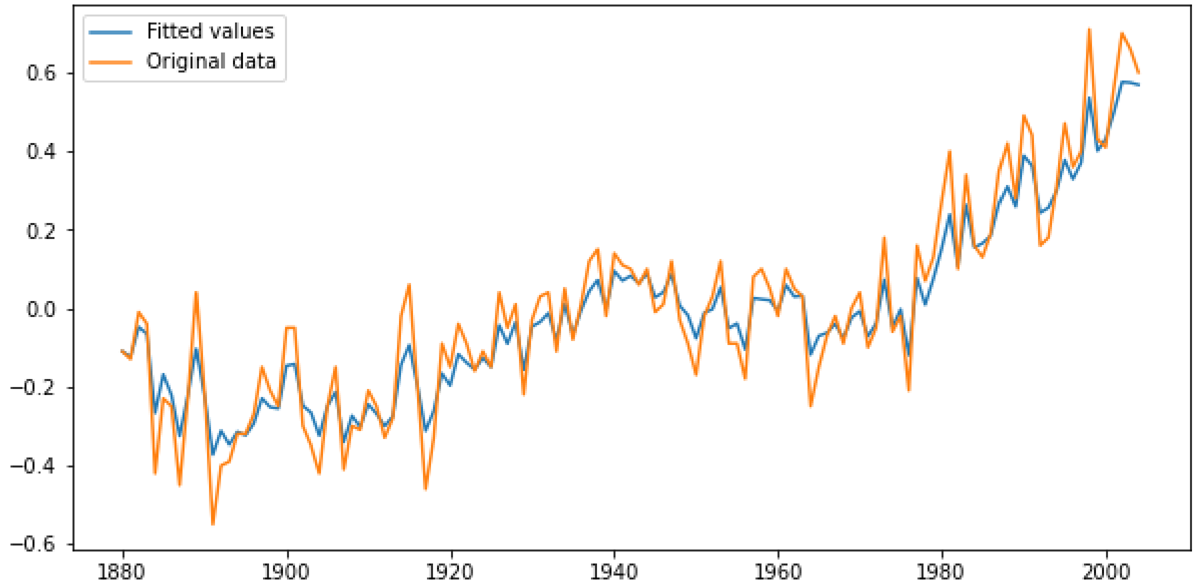
	coef	std err	z	P>|z|	[0.025	0.975]
ma.L1	-0.4676	0.099	-4.736	0.000	-0.661	-0.274
ma.L2	-0.2296	0.090	-2.565	0.010	-0.405	-0.054
sigma2	0.0170	0.002	7.631	0.000	0.013	0.021

Ljung-Box (L1) (Q):	0.13	**Jarque-Bera (JB):**	1.18
Prob(Q):	0.72	**Prob(JB):**	0.55
Heteroskedasticity (H):	1.04	**Skew:**	-0.23
Prob(H) (two-sided):	0.91	**Kurtosis:**	3.12

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [37]:

```
df_pred = model_fit.predict(start=1,end=len(df['Anomaly, C']), typ = 'levels').rename("Predictions")
plt.plot (df.index, df_pred, label='Fitted values')
plt.plot (df.index, df['Anomaly, C'], label='Original data')
plt.legend()
plt.show()
```



In [38]:
```
sse = mean_squared_error (df['Anomaly, C'].values, df_pred.values) * len(df_pred)
sse,len(df_pred.values)
```

Out[38]:
```
(0.5617319358156067, 125)
```

Part c

Comment whether model (a) or (b) is better suited for this data based on the SSE? (3 pts)

Eventhough the SSE of IMA(1,1) is higher than that of IMA(1,2) it can be seen that IMA(1,2) has a tendency to overfit. Since IMA(1,1) can generalize the data better, IMA(1,1) would be ther better choice.

Question 3

Review the dataset in the file Measurement_Q3.xls which contains measurements recorded annually over close to 50 years. (15 pts)

In [39]:
```
df = pd.read_csv('Measurement_Q3.csv', index_col = 'Year')
df
```

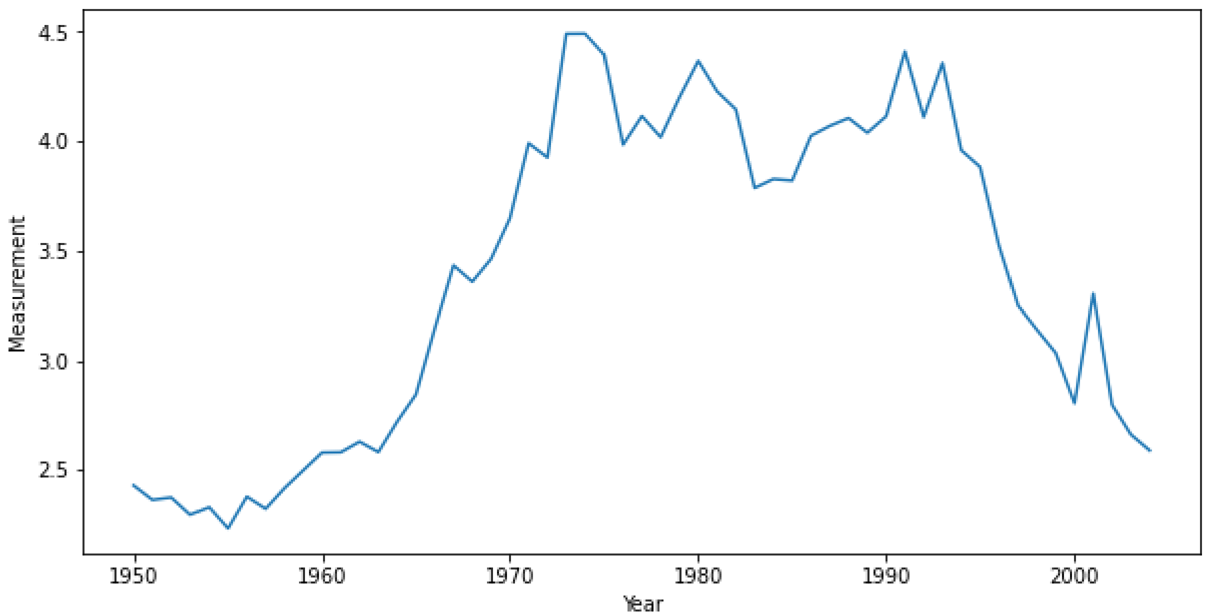
Out[39]:

	Measurement
Year	
1950	2.429415
1951	2.363364
1952	2.374305
1953	2.295520
1954	2.329716
1955	2.233017
1956	2.378179
1957	2.322671
1958	2.416556
1959	2.498199
1960	2.579453
1961	2.580840
1962	2.629293
1963	2.581853
1964	2.720940
1965	2.844774
1966	3.144862
1967	3.433044
1968	3.358418
1969	3.462620
1970	3.647342
1971	3.991080
1972	3.925702
1973	4.490962
1974	4.491541

	Measurement
Year	
1975	4.396567
1976	3.984491
1977	4.115111
1978	4.018538
1979	4.201107
1980	4.367459
1981	4.228103
1982	4.145889
1983	3.786691
1984	3.827373
1985	3.820376
1986	4.025134
1987	4.070130
1988	4.105920
1989	4.039027
1990	4.113978
1991	4.410670
1992	4.110586
1993	4.357700
1994	3.959040
1995	3.882907
1996	3.524803
1997	3.249564
1998	3.139884
1999	3.034263

	Measurement
Year	
2000	2.805041
2001	3.304467
2002	2.797697
2003	2.662227
2004	2.589383

In [40]:
```
plt.xlabel('Year')
plt.ylabel('Measurement')
plt.plot(df)
plt.show()
```

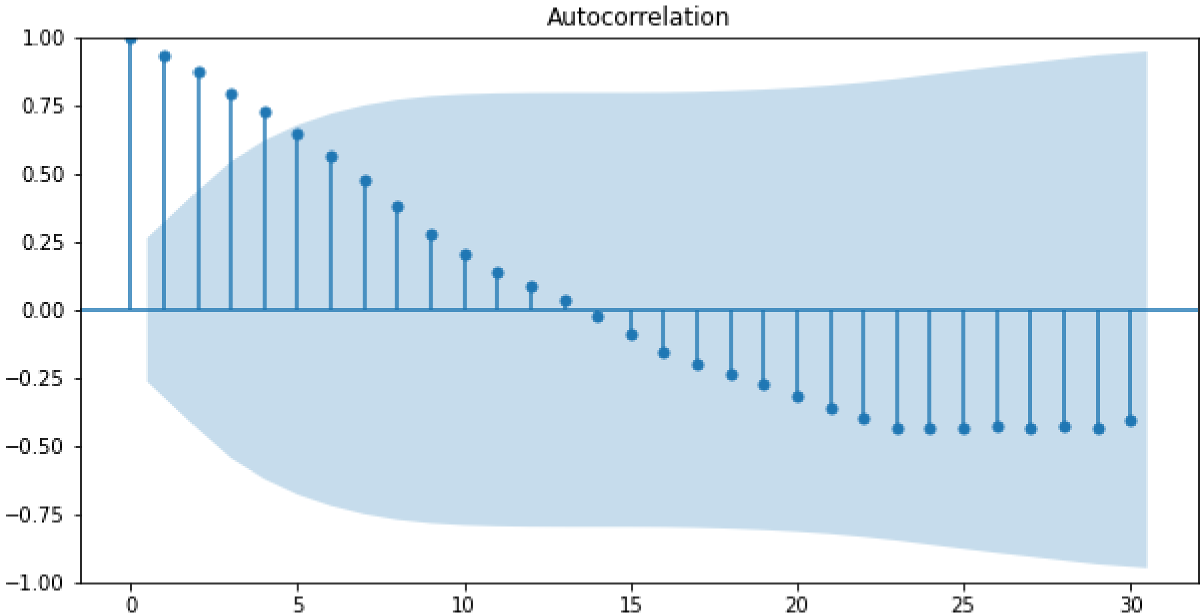


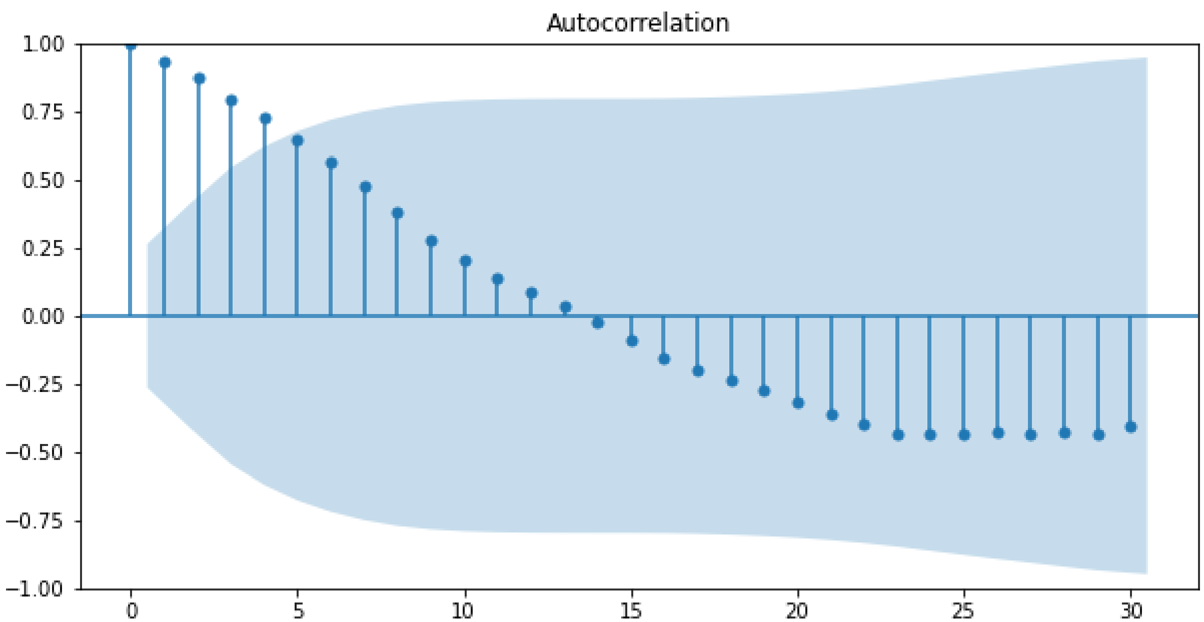
Part a

Plot the time series, ACF, and Partial AutoCorrelation Function (PACF). (6 pts)

In [43]:
```
plot_acf(df.Measurement, lags = 30)
```

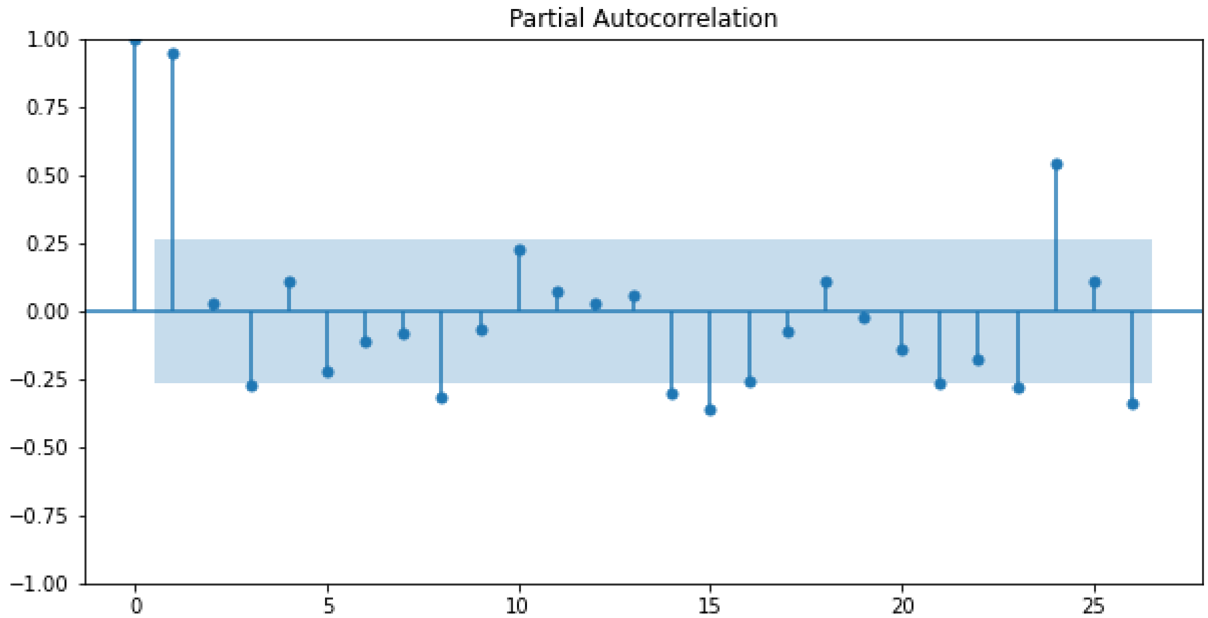
Out[43]:

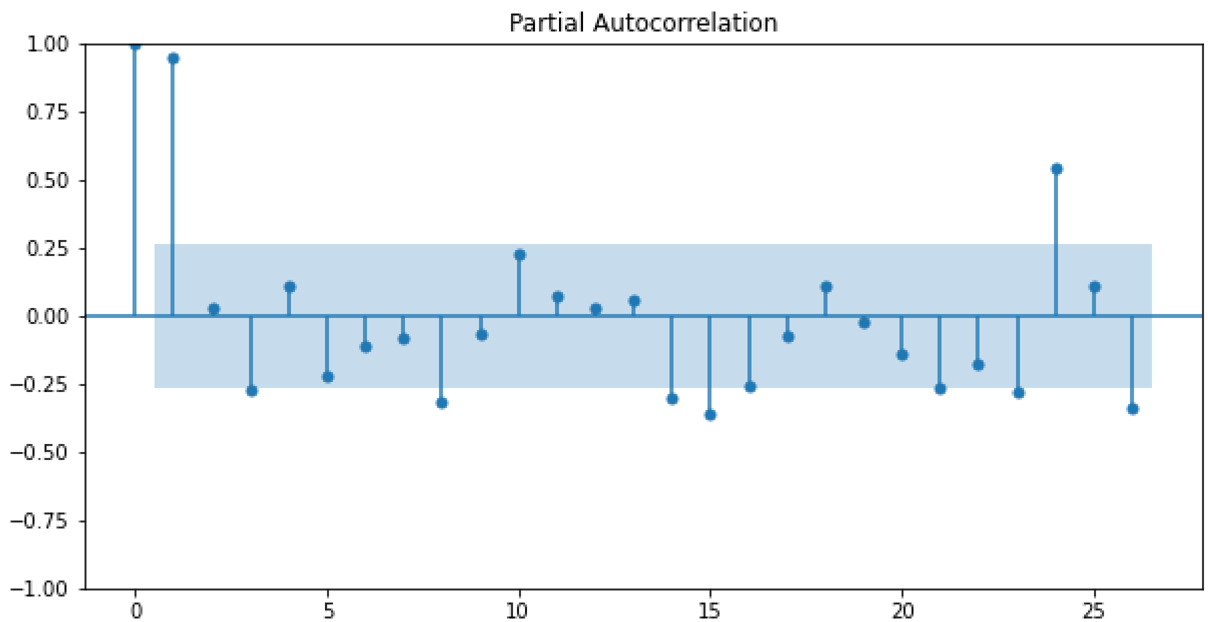




In [45]:
```
plot_pacf(df.Measurement, lags = 26)
```

Out[45]:



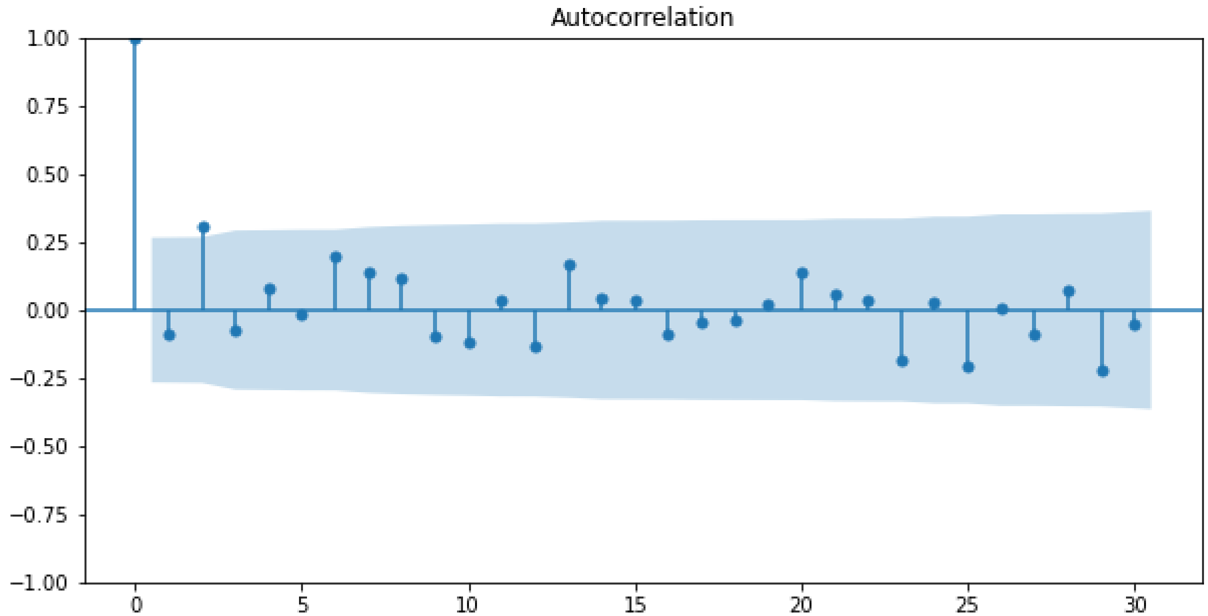


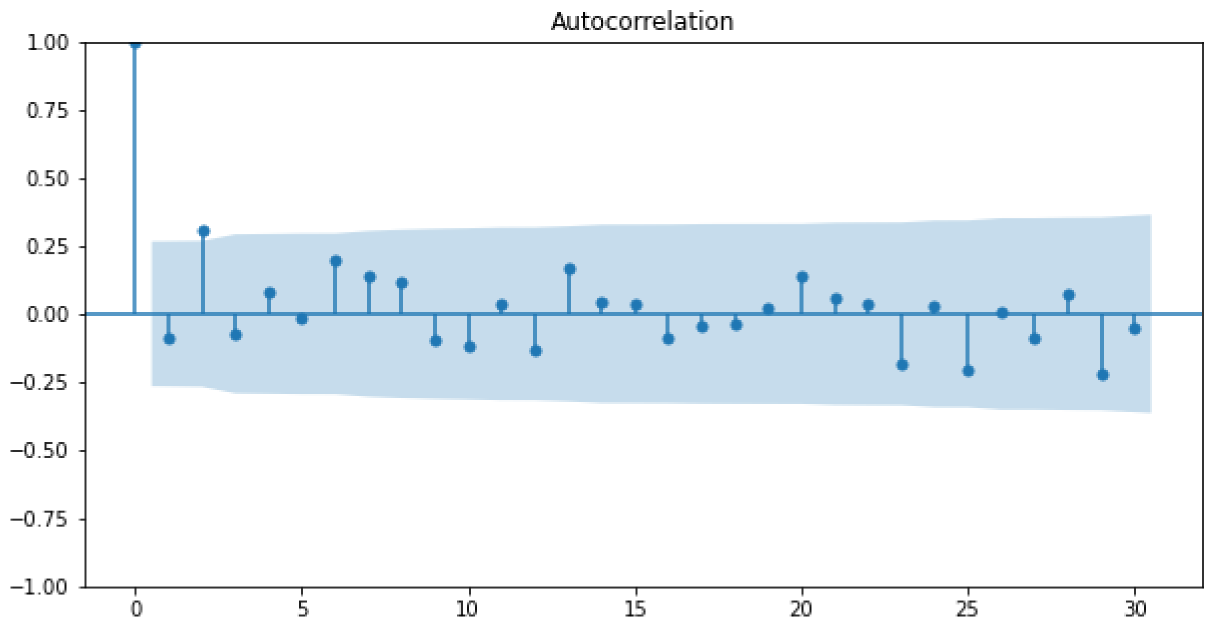
Part b

Compute the first difference and plot its ACF and PACF. (6 pts)

In [48]:
```
plot_acf(df.diff().dropna(), lags = 30)
```

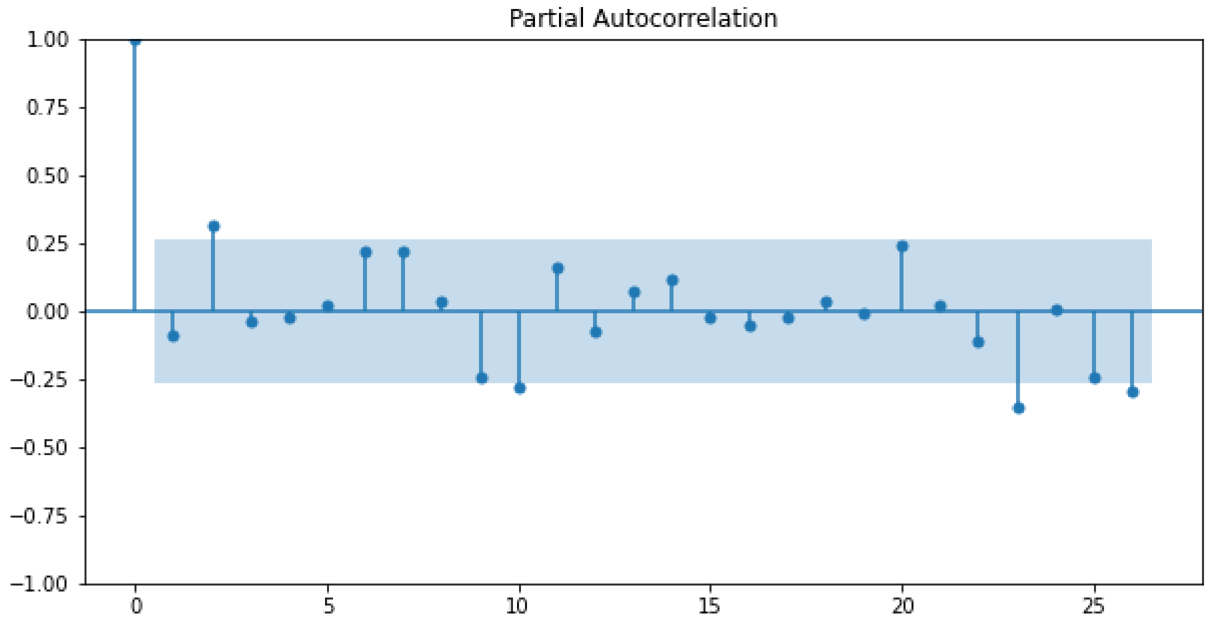
Out[48]:

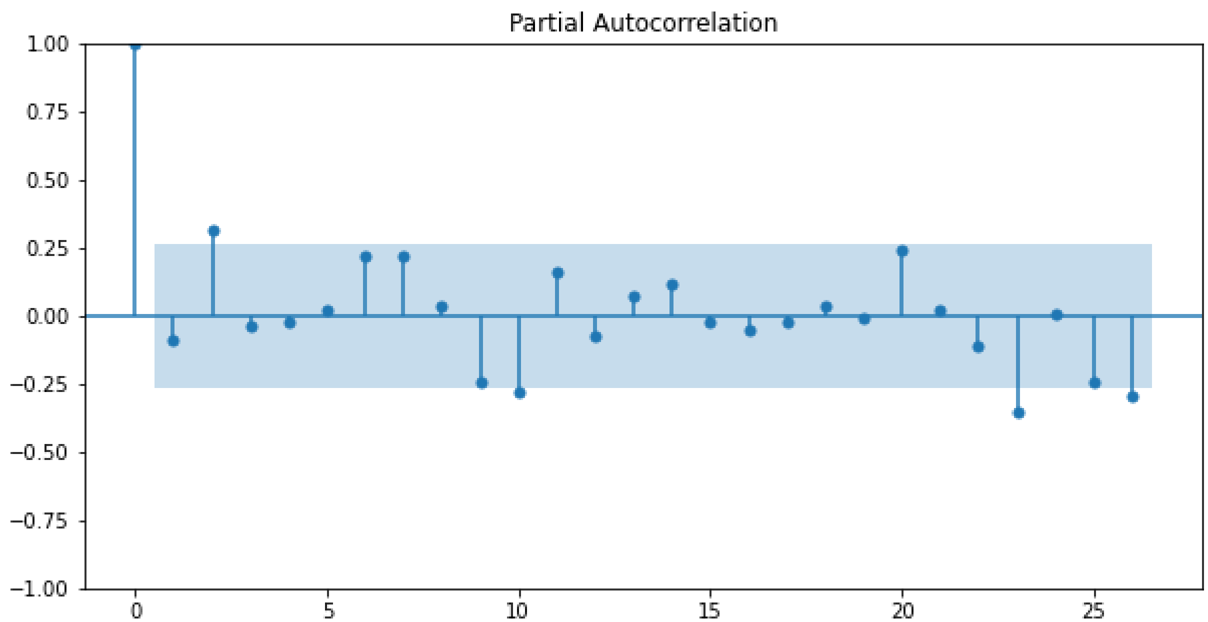




In [49]:
```
plot_pacf(df.diff().dropna(), lags = 26)
```

Out[49]:


Partial Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25


Partial Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25

Part c

Based on the results in (b) above, what model order, i.e. p,d,q in ARMA(p,d,q), would you recommend for the above time series? Provide a justification for your answer. (3 pts)

According tp the plots above d should be equal to 1 and the values of p and q can be 0, 1 or 2. So, it might be worth fitting different combinations of these values to multiple ARIMA models and pick the one with the lowest AIC Score.

In [57]:
```
# Arima(0,1,1)

arima = ARIMA(df.Measurement, order=(0,1,1))
arima_fit = arima.fit()
arima_fit.summary()
```

Out[57]:

SARIMAX Results

Dep. Variable:	Measurement	**No. Observations:**	55
Model:	ARIMA(0, 1, 1)	**Log Likelihood**	6.850
Date:	Thu, 13 Oct 2022	**AIC**	-9.700
Time:	13:42:19	**BIC**	-5.722
Sample:	0	**HQIC**	-8.166
	- 55		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ma.L1	-0.0527	0.130	-0.404	0.686	-0.308	0.203
sigma2	0.0454	0.009	5.320	0.000	0.029	0.062

Ljung-Box (L1) (Q):	0.02	**Jarque-Bera (JB):**	0.23
Prob(Q):	0.90	**Prob(JB):**	0.89
Heteroskedasticity (H):	4.05	**Skew:**	0.07
Prob(H) (two-sided):	0.00	**Kurtosis:**	3.29

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [56]:

```
# Arima(0,1,2)

arima = ARIMA(df.Measurement, order=(0,1,2))
arima_fit = arima.fit()
arima_fit.summary()
```

Out[56]:

SARIMAX Results

Dep. Variable:	Measurement	**No. Observations:**	55
Model:	ARIMA(0, 1, 2)	**Log Likelihood**	9.713
Date:	Thu, 13 Oct 2022	**AIC**	-13.425
Time:	13:41:57	**BIC**	-7.458
Sample:	0	**HQIC**	-11.124
	- 55		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ma.L1	-0.0660	0.110	-0.598	0.550	-0.282	0.150
ma.L2	0.3712	0.128	2.907	0.004	0.121	0.622
sigma2	0.0406	0.008	4.999	0.000	0.025	0.057

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	0.03
Prob(Q):	0.96	**Prob(JB):**	0.99
Heteroskedasticity (H):	4.20	**Skew:**	-0.00
Prob(H) (two-sided):	0.00	**Kurtosis:**	2.89

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [55]:

```
# Arima(1,1,0)

arima = ARIMA(df.Measurement, order=(1,1,0))
arima_fit = arima.fit()
arima_fit.summary()
```

Out[55]:

SARIMAX Results

Dep. Variable:	Measurement	**No. Observations:**	55
Model:	ARIMA(1, 1, 0)	**Log Likelihood**	6.923
Date:	Thu, 13 Oct 2022	**AIC**	-9.846
Time:	13:41:38	**BIC**	-5.868
Sample:	0	**HQIC**	-8.312
	- 55		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.0841	0.129	-0.653	0.514	-0.336	0.168
sigma2	0.0453	0.009	5.238	0.000	0.028	0.062

Ljung-Box (L1) (Q):	0.03	**Jarque-Bera (JB):**	0.17
Prob(Q):	0.86	**Prob(JB):**	0.92
Heteroskedasticity (H):	3.92	**Skew:**	0.08
Prob(H) (two-sided):	0.01	**Kurtosis:**	3.23

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [59]:

```
# Arima(2,1,0)

arima = ARIMA(df.Measurement, order=(2,1,0))
arima_fit = arima.fit()
arima_fit.summary()
```

Out[59]:

SARIMAX Results

Dep. Variable:	Measurement	**No. Observations:**	55
Model:	ARIMA(2, 1, 0)	**Log Likelihood**	9.479
Date:	Thu, 13 Oct 2022	**AIC**	-12.959
Time:	13:42:37	**BIC**	-6.992
Sample:	0	**HQIC**	-10.657
	- 55		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.0572	0.118	-0.486	0.627	-0.287	0.173
ar.L2	0.2967	0.125	2.371	0.018	0.051	0.542
sigma2	0.0411	0.008	4.965	0.000	0.025	0.057

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	0.06
Prob(Q):	0.96	**Prob(JB):**	0.97
Heteroskedasticity (H):	4.24	**Skew:**	0.08
Prob(H) (two-sided):	0.00	**Kurtosis:**	2.96

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [60]:

```
# Arima(1,1,1)

arima = ARIMA(df.Measurement, order=(1,1,1))
arima_fit = arima.fit()
arima_fit.summary()
```

Out[60]:

SARIMAX Results

Dep. Variable:	Measurement	**No. Observations:**	55
Model:	ARIMA(1, 1, 1)	**Log Likelihood**	8.178
Date:	Thu, 13 Oct 2022	**AIC**	-10.357
Time:	13:43:17	**BIC**	-4.390
Sample:	0	**HQIC**	-8.056
	- 55		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.8254	0.326	-2.530	0.011	-1.465	-0.186
ma.L1	0.6915	0.408	1.696	0.090	-0.108	1.491
sigma2	0.0432	0.009	4.977	0.000	0.026	0.060

Ljung-Box (L1) (Q):	0.66	**Jarque-Bera (JB):**	0.10
Prob(Q):	0.42	**Prob(JB):**	0.95
Heteroskedasticity (H):	3.81	**Skew:**	0.11
Prob(H) (two-sided):	0.01	**Kurtosis:**	2.98

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [61]:

```
# Arima(2,1,2)

arima = ARIMA(df.Measurement, order=(2,1,2))
arima_fit = arima.fit()
arima_fit.summary()
```

Out[61]:

SARIMAX Results

Dep. Variable:	Measurement	**No. Observations:**	55
Model:	ARIMA(2, 1, 2)	**Log Likelihood**	11.857
Date:	Thu, 13 Oct 2022	**AIC**	-13.714
Time:	13:43:35	**BIC**	-3.769
Sample:	0	**HQIC**	-9.879
	- 55		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.3859	0.128	3.025	0.002	0.136	0.636
ar.L2	-0.5070	0.178	-2.847	0.004	-0.856	-0.158
ma.L1	-0.4354	6.479	-0.067	0.946	-13.134	12.263
ma.L2	0.9996	29.774	0.034	0.973	-57.357	59.356
sigma2	0.0347	1.030	0.034	0.973	-1.985	2.054

Ljung-Box (L1) (Q):	0.23	**Jarque-Bera (JB):**	0.48
Prob(Q):	0.64	**Prob(JB):**	0.79
Heteroskedasticity (H):	4.36	**Skew:**	-0.23
Prob(H) (two-sided):	0.00	**Kurtosis:**	3.09

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

The lollipop graph suggest p=2 and q=2 which is confirmed by the low AIC Score for ARIMA(2,1,2).

In [ ]: