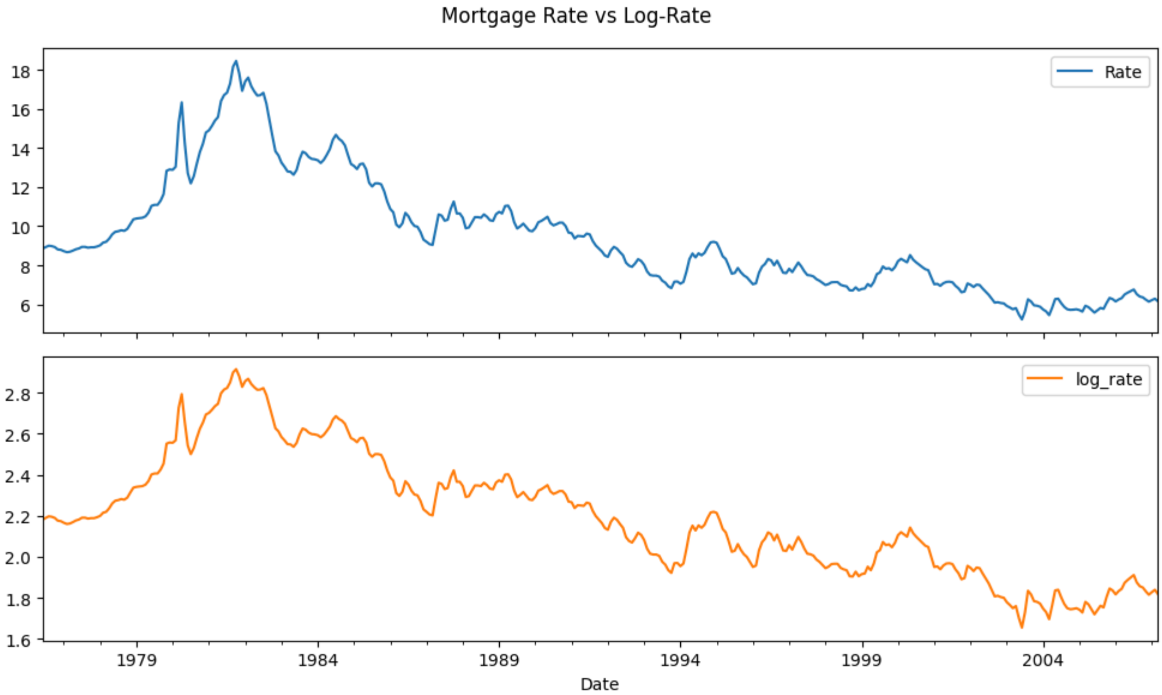
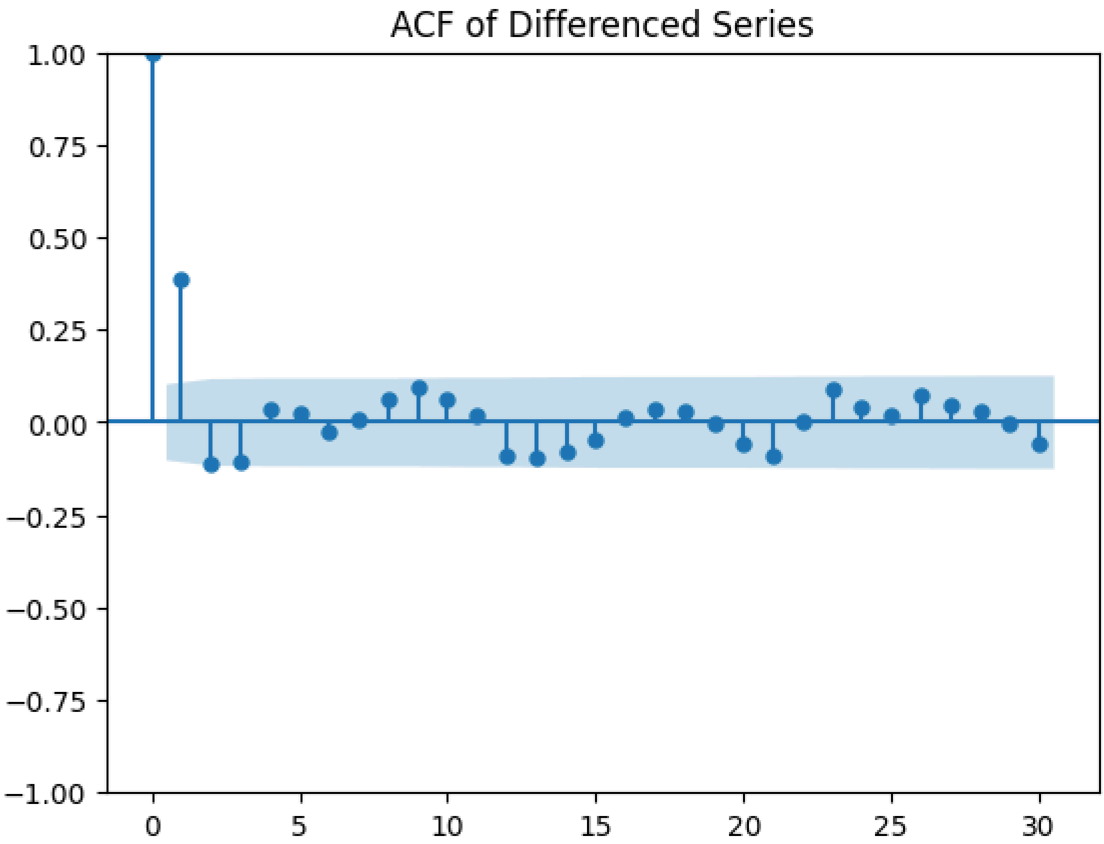
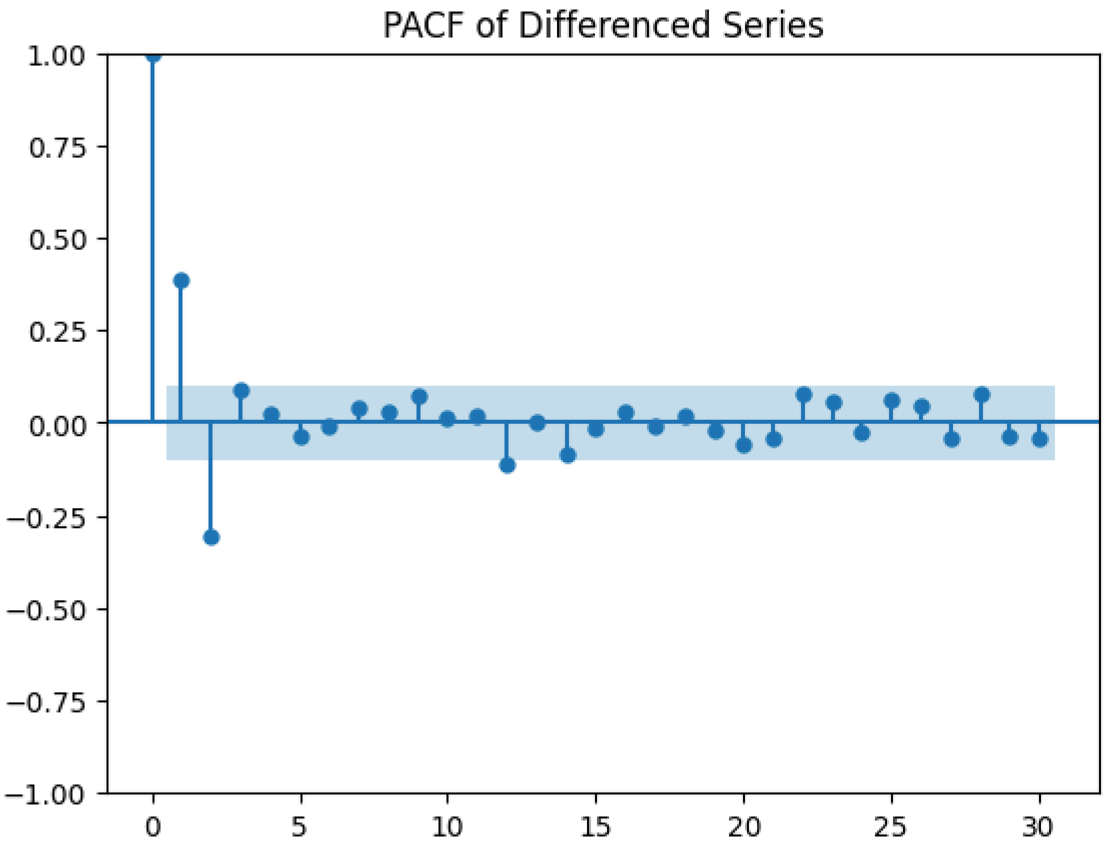
In [1]:
```
# Time Series Modeling for 30-Year U.S. Mortgage Rate

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.stattools import adfuller
from statsmodels.stats.diagnostic import acorr_ljungbox
import warnings
warnings.filterwarnings("ignore")

# Step 1: Load and preprocess data
df = pd.read_csv("m-mortg.txt", sep="\s+", header=None, names=["Year", "M
df["Date"] = pd.to_datetime(df[["Year", "Month", "Day"]])
df.set_index("Date", inplace=True)
df = df[["Rate"]]
df["log_rate"] = np.log(df["Rate"])

# Step 2: Visualize original and log-transformed data
df[["Rate", "log_rate"]].plot(title="Mortgage Rate vs Log-Rate", subplots
plt.tight_layout()
plt.show()

# Step 3: ADF test for stationarity
result = adfuller(df['log_rate'])
print("ADF Test p-value:", result[1])

# Step 4: First difference the series
df['log_rate_diff'] = df['log_rate'].diff()

# Step 5: ACF and PACF plots
plot_acf(df['log_rate_diff'].dropna(), lags=30)
plt.title("ACF of Differenced Series")
plt.show()
plot_pacf(df['log_rate_diff'].dropna(), lags=30)
plt.title("PACF of Differenced Series")
plt.show()

# Step 6: Auto ARIMA grid search
def auto_arima_grid_search(series, d, p_range=range(0, 4), q_range=range(
    best_aic = np.inf
    best_order = None
    best_model = None
    for p in p_range:
        for q in q_range:
            try:
                model = ARIMA(series, order=(p, d, q))
                result = model.fit()
                if result.aic < best_aic:
                    best_aic = result.aic
                    best_order = (p, d, q)
                    best_model = result
            except:
                continue
    return best_order, best_model

best_order, best_model = auto_arima_grid_search(df["log_rate"], d=1)
```

```
print("Best ARIMA model:", best_order)
print(best_model.summary())

# Step 7: Model diagnostics
resid = best_model.resid
lb_test = acorr_ljungbox(resid, lags=[12], return_df=True)
print("\nLjung-Box Q(12) Test:")
print(lb_test)

# Step 8: Forecast 1-4 steps ahead
forecast = best_model.get_forecast(steps=4)
log_forecast = forecast.predicted_mean
rate_forecast = np.exp(log_forecast)

# Step 9: Display forecast results
print("\n1 to 4-step forecast (original rate):")
for i in range(4):
    print(f"Step {i+1}: {rate_forecast.iloc[i]:.4f}")
```




```
ADF Test p-value: 0.8196642809000326
```


ACF of Differenced Series
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
30


PACF of Differenced Series
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
30

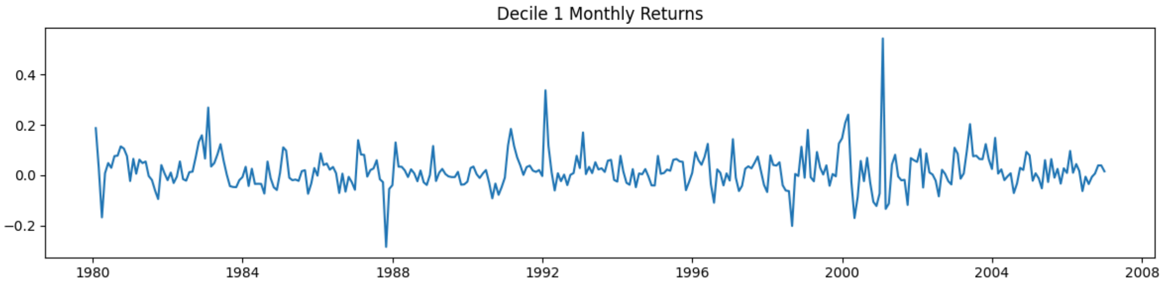
```
Best ARIMA model: (3, 1, 0)
                               SARIMAX Results
==========================================================================
====
Dep. Variable:                log_rate   No. Observations:
370
Model:                  ARIMA(3, 1, 0)   Log Likelihood                 82
0.726
Date:                 Sat, 05 Apr 2025   AIC                          -163
3.453
Time:                         13:23:24   BIC                          -161
7.810
Sample:                     06-01-1976   HQIC                         -162
7.239
                          - 03-01-2007
Covariance Type:                   opg
==========================================================================
====
                 coef    std err          z      P>|z|      [0.025      0.
975]
--------------------------------------------------------------------------
----
ar.L1          0.5323      0.049     10.758      0.000       0.435
0.629
ar.L2         -0.3491      0.041     -8.560      0.000      -0.429      -
0.269
ar.L3          0.0894      0.054      1.654      0.098      -0.017
0.195
sigma2         0.0007   3.12e-05     21.949      0.000       0.001
0.001
==========================================================================
=========
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):
281.52
Prob(Q):                              0.96   Prob(JB):
0.00
Heteroskedasticity (H):               0.92   Skew:
0.67
Prob(H) (two-sided):                  0.64   Kurtosis:
7.07
==========================================================================
=========

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

Ljung-Box Q(12) Test:
     lb_stat  lb_pvalue
12  0.05655        1.0

1 to 4-step forecast (original rate):
Step 1: 6.0752
Step 2: 6.0807
Step 3: 6.1018
Step 4: 6.1035
```

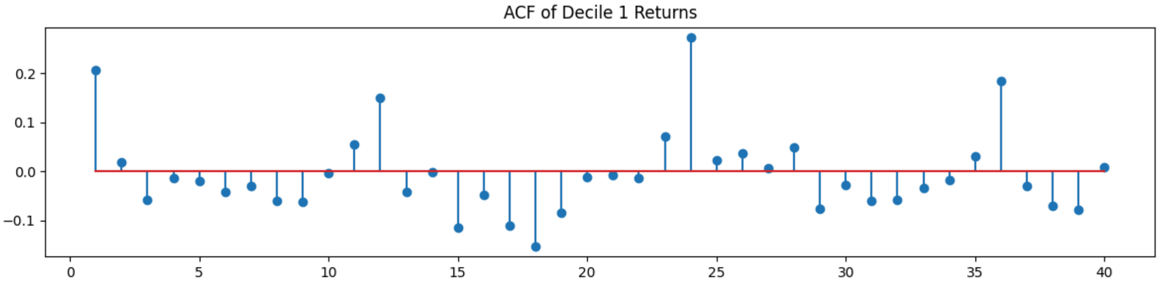
In [2]:

```python
# Time Series Modeling for Decile 1 Monthly Returns using SARIMA

import pandas as pd
```

```
import numpy as np
from statsmodels.tsa.statespace.sarimax import SARIMAX
import matplotlib.pyplot as plt
from statsmodels.tsa.stattools import acf
from statsmodels.stats.diagnostic import acorr_ljungbox
from scipy import stats

# Step 1: Load data
dec1 = pd.read_csv('m-dec1-8006.txt', sep='\s+', header=None)
dec1.columns = ['date', 'return']

# Step 2: Create datetime index
dates = pd.date_range(start='1980-01-01', periods=len(dec1), freq='M')
dec1_ts = pd.Series(dec1['return'].values, index=dates)

# Step 3: Plot time series and ACF
plt.figure(figsize=(12, 6))
plt.subplot(211)
plt.plot(dec1_ts)
plt.title('Decile 1 Monthly Returns')
plt.subplot(212)
acf_values = acf(dec1_ts.values, nlags=40)[1:]
plt.stem(range(1, len(acf_values) + 1), acf_values)
plt.title('ACF of Decile 1 Returns')
plt.tight_layout()
plt.show()

# Step 4: Fit SARIMA(0,0,1)(1,0,1)[12] model
model = SARIMAX(dec1_ts, order=(0, 0, 1), seasonal_order=(1, 0, 1, 12))
results = model.fit()
print(results.summary())

# Step 5: Residual diagnostics
residuals = results.resid
plt.figure(figsize=(12, 8))
plt.subplot(221)
plt.plot(residuals)
plt.title('Residuals Over Time')
plt.subplot(222)
plt.hist(residuals, bins=30)
plt.title('Histogram of Residuals')
plt.subplot(223)
acf_values = acf(residuals.values, nlags=40)[1:]
plt.stem(range(1, len(acf_values) + 1), acf_values)
plt.title('ACF of Residuals')
plt.subplot(224)
stats.probplot(residuals, dist="norm", plot=plt)
plt.title('Q-Q Plot of Residuals')
plt.tight_layout()
plt.show()

# Step 6: Ljung-Box test (Q(24))
lb_result = acorr_ljungbox(residuals, lags=[24], return_df=True)
q_stat = lb_result['lb_stat'].iloc[0]
p_val = lb_result['lb_pvalue'].iloc[0]
print(f"Ljung-Box Q(24) Statistic: {q_stat:.4f}")
print(f"p-value: {p_val:.4f}")

# Step 7: Conclusion
alpha = 0.05
```

```
if p_val > alpha:
    print(f"Residuals show no significant autocorrelation (p > {alpha}).
else:
    print(f"Residuals show significant autocorrelation (p ≤ {alpha}). Mod
```

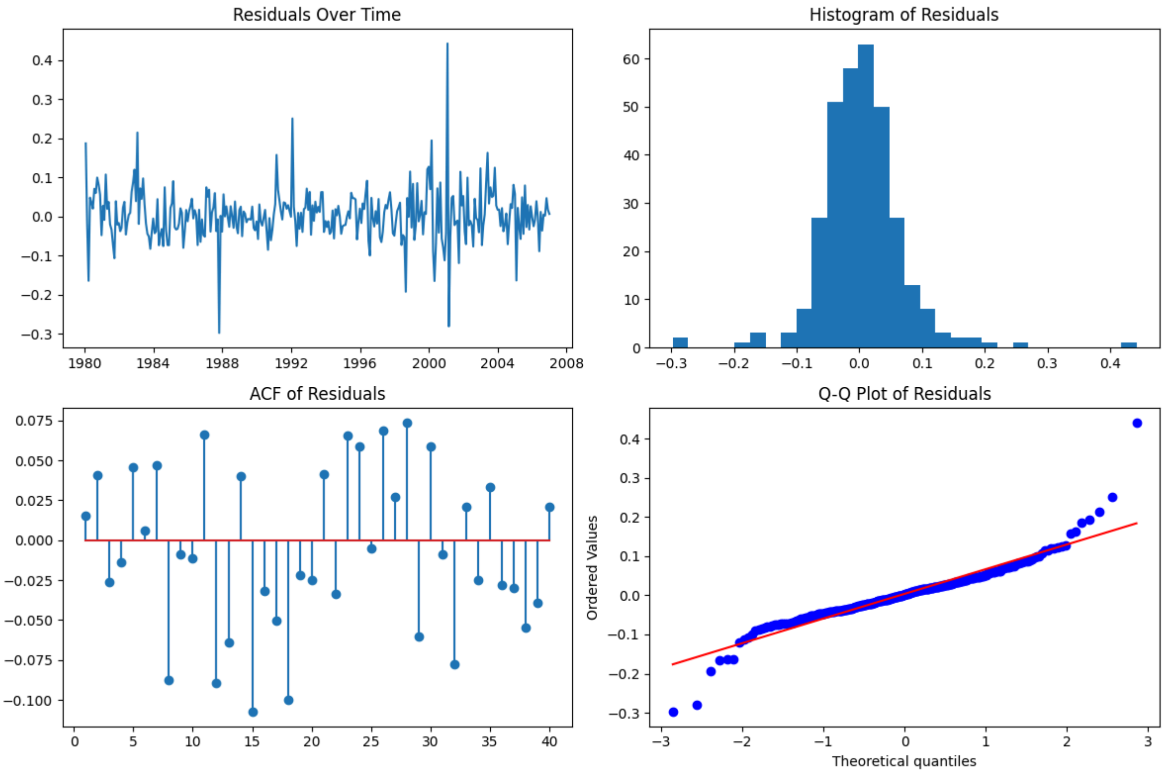





```
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            4     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f= -1.22381D+00    |proj g|=  6.68059D-01

At iterate    5    f= -1.22788D+00    |proj g|=  1.03522D+00

At iterate   10    f= -1.23114D+00    |proj g|=  8.61734D-02

At iterate   15    f= -1.23382D+00    |proj g|=  1.13082D+00

At iterate   20    f= -1.25902D+00    |proj g|=  2.80640D-01
 This problem is unconstrained.
```

```
At iterate   25    f= -1.28620D+00    |proj g|=  1.53820D-01

At iterate   30    f= -1.29205D+00    |proj g|=  2.60927D-01

At iterate   35    f= -1.29438D+00    |proj g|=  1.65326D-01

At iterate   40    f= -1.29491D+00    |proj g|=  2.71945D-02

At iterate   45    f= -1.29502D+00    |proj g|=  1.48042D-03

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    4     47     59      1     0     0   1.412D-03  -1.295D+00
  F =  -1.2950159735225941

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH
                                     SARIMAX Results
==========================================================================
================
Dep. Variable:                                 y   No. Observations:
324
Model:             SARIMAX(0, 0, 1)x(1, 0, 1, 12)   Log Likelihood
419.585
Date:                            Sat, 05 Apr 2025   AIC
-831.170
Time:                                    13:23:26   BIC
-816.047
Sample:                                01-31-1980   HQIC
-825.134
                                     - 12-31-2006
Covariance Type:                              opg
==========================================================================
====
                 coef    std err          z      P>|z|      [0.025      0.
975]
--------------------------------------------------------------------------
----
ma.L1          0.2455      0.034      7.196      0.000       0.179
0.312
ar.S.L12       0.9997      0.012     85.199      0.000       0.977
1.023
ma.S.L12      -0.9862      0.293     -3.366      0.001      -1.561      -
0.412
sigma2         0.0041      0.001      4.008      0.000       0.002
0.006
==========================================================================
=========
Ljung-Box (L1) (Q):                   0.02   Jarque-Bera (JB):
1059.98
```

```
Prob(Q):                              0.89   Prob(JB):
0.00
Heteroskedasticity (H):               1.93   Skew:
0.69
Prob(H) (two-sided):                  0.00   Kurtosis:
11.75
===========================================================================
=========

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




```
Ljung-Box Q(24) Statistic: 23.7876
p-value: 0.4738
Residuals show no significant autocorrelation (p > 0.05). Model is adequat
e.
```

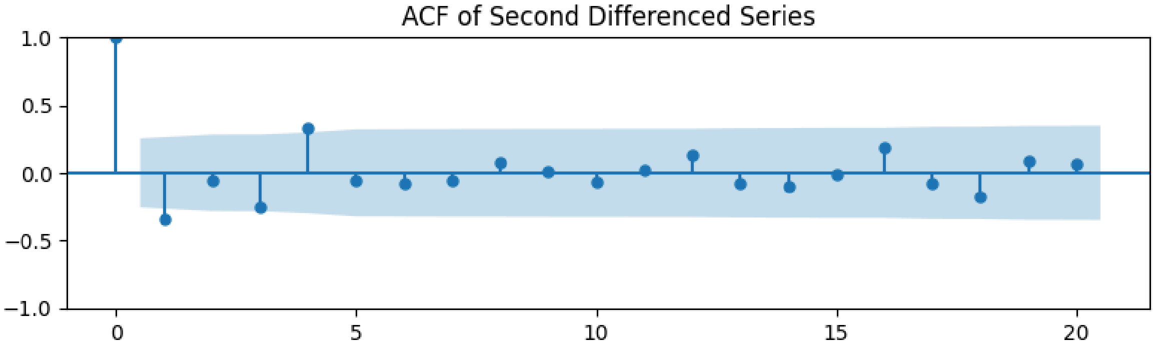
In [3]:

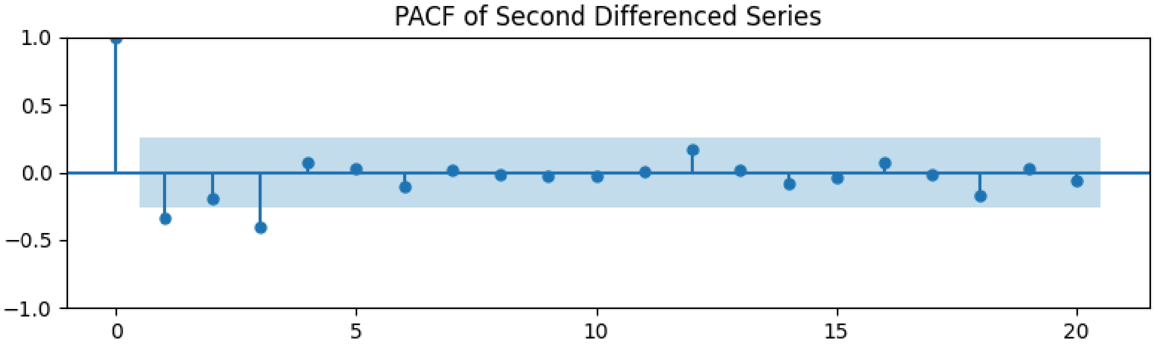
```
# Alcoa Quarterly EPS Time Series Modeling and Forecasting

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import itertools
import warnings
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.stattools import adfuller
from statsmodels.stats.diagnostic import acorr_ljungbox

warnings.filterwarnings("ignore")

# Step 1: Load and preprocess data
df = pd.read_csv("q-aa-earn.txt", sep="\s+", header=None, names=["Day", "
df["Date"] = pd.to_datetime(df[["Year", "Month", "Day"]])
df.set_index("Date", inplace=True)
df = df.sort_index()
eps = df["EPS"]
```

```
# Step 2: Stationarity check (ADF Test)
result = adfuller(eps)
print("ADF p-value (original series):", result[1])

# Step 3: Apply second-order differencing (since series is non-stationary
diff_eps = eps.diff().dropna().diff().dropna()
result_diff2 = adfuller(diff_eps)
print("ADF p-value (second difference):", result_diff2[1])

# Step 4: Plot ACF and PACF for second-differenced series
fig, axes = plt.subplots(2, 1, figsize=(8, 5))
plot_acf(diff_eps, lags=20, ax=axes[0])
axes[0].set_title('ACF of Second Differenced Series')
plot_pacf(diff_eps, lags=20, ax=axes[1])
axes[1].set_title('PACF of Second Differenced Series')
plt.tight_layout()
plt.show()

# Step 5: Try multiple ARIMA models and select based on AIC
candidate_orders = [(0, 2, 1), (1, 2, 0), (1, 2, 1), (2, 2, 0), (0, 2, 2)
results = []
for order in candidate_orders:
    model = ARIMA(eps, order=order)
    fitted_model = model.fit()
    results.append((order, fitted_model))
    print(f"ARIMA{order} - AIC: {fitted_model.aic:.2f}")

best_order, best_model = min(results, key=lambda x: x[1].aic)
print(f"\nBest ARIMA model: {best_order} (AIC = {best_model.aic:.2f})")
print(best_model.summary())

# Step 6: Residual diagnostics
resid = best_model.resid
lb_result = acorr_ljungbox(resid, lags=[12], return_df=True)
print("\nLjung-Box Q(12) Test:")
print(lb_result)

# Step 7: Forecast next 4 quarters
forecast = best_model.get_forecast(steps=4)
mean_forecast = forecast.predicted_mean
conf_int = forecast.conf_int()

# Step 8: Plot forecast
last_date = df.index[-1]
future_dates = pd.date_range(start=last_date + pd.DateOffset(months=3), p

plt.figure(figsize=(10, 6))
plt.plot(df.index, df['EPS'], label='Historical EPS')
plt.plot(future_dates, mean_forecast, color='red', label='Forecast')
plt.fill_between(future_dates, conf_int.iloc[:, 0], conf_int.iloc[:, 1],
plt.title('Alcoa Quarterly EPS Forecast')
plt.xlabel('Date')
plt.ylabel('EPS')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

# Step 9: Print forecast values
```

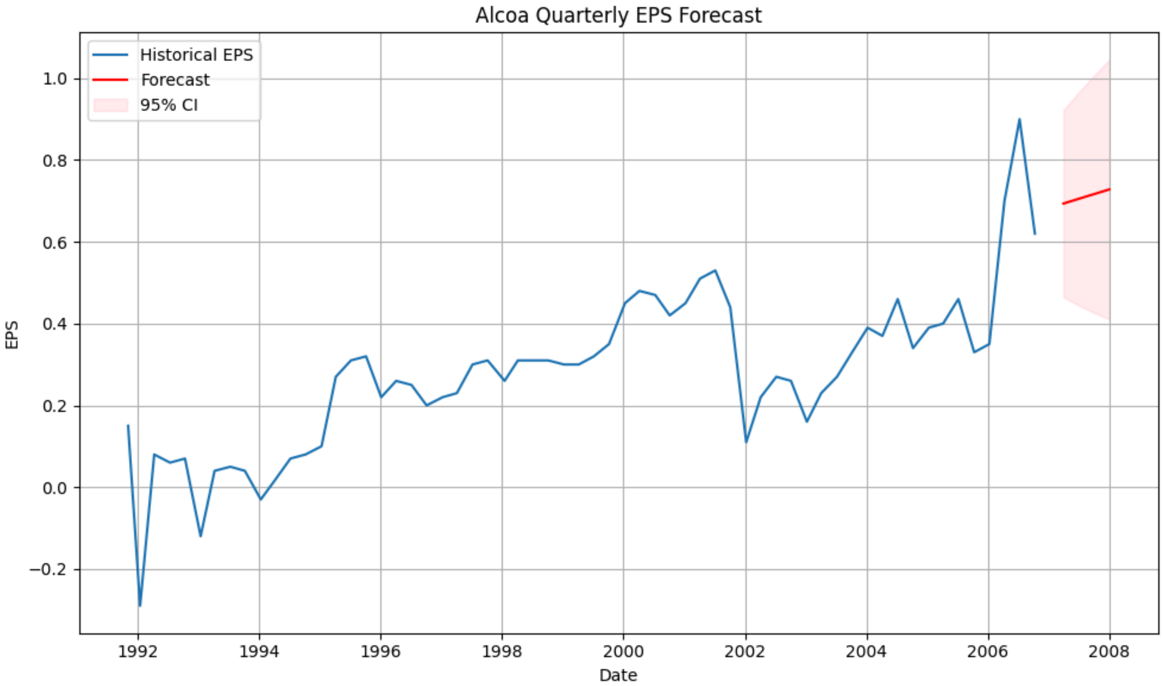
```
print("\nForecast Summary:")
for i in range(4):
    print(f"Quarter {i+1}: {mean_forecast.iloc[i]:.4f}, 95% CI: [{conf_in
```

```
ADF p-value (original series): 0.6181931025603755
ADF p-value (second difference): 4.894164446594406e-22
```





```
ARIMA(0, 2, 1) - AIC: -70.09
ARIMA(1, 2, 0) - AIC: -42.49
ARIMA(1, 2, 1) - AIC: -72.07
ARIMA(2, 2, 0) - AIC: -47.25
ARIMA(0, 2, 2) - AIC: -75.51
ARIMA(2, 2, 2) - AIC: -73.68

Best ARIMA model: (0, 2, 2) (AIC = -75.51)
                               SARIMAX Results                                
==========================================================================
====
Dep. Variable:                    EPS   No. Observations:                 
61
Model:                 ARIMA(0, 2, 2)   Log Likelihood                  4
0.753
Date:                Sat, 05 Apr 2025   AIC                            -7
5.506
Time:                        13:23:27   BIC                            -6
9.274
Sample:                             0   HQIC                           -7
3.073
                                 - 61                                         
Covariance Type:                  opg                                         
==========================================================================
====
                 coef    std err          z      P>|z|      [0.025      0.
975]
--------------------------------------------------------------------------
----
ma.L1         -1.4639      0.467     -3.137      0.002      -2.378      -
0.549
ma.L2          0.4688      0.175      2.686      0.007       0.127       
0.811
sigma2         0.0135      0.007      2.045      0.041       0.001       
0.026
==========================================================================
=========
Ljung-Box (L1) (Q):                   3.04   Jarque-Bera (JB):             
63.12
Prob(Q):                              0.08   Prob(JB):                      
0.00
Heteroskedasticity (H):               1.44   Skew:                          
0.76
Prob(H) (two-sided):                  0.42   Kurtosis:                      
7.83
==========================================================================
=========

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

Ljung-Box Q(12) Test:
     lb_stat  lb_pvalue
12  7.956544   0.788517
```




```
Forecast Summary:
Quarter 1: 0.6937, 95% CI: [0.4650, 0.9224]
Quarter 2: 0.7051, 95% CI: [0.4447, 0.9656]
Quarter 3: 0.7166, 95% CI: [0.4269, 1.0063]
Quarter 4: 0.7281, 95% CI: [0.4110, 1.0452]
```

In [ ]:

In [ ]: