DATS 6313 – Time Series Analysis & Modeling

Instructor: Reza Jafari

Lab #8

Bradley Reardon

4/13/2022

Abstract:

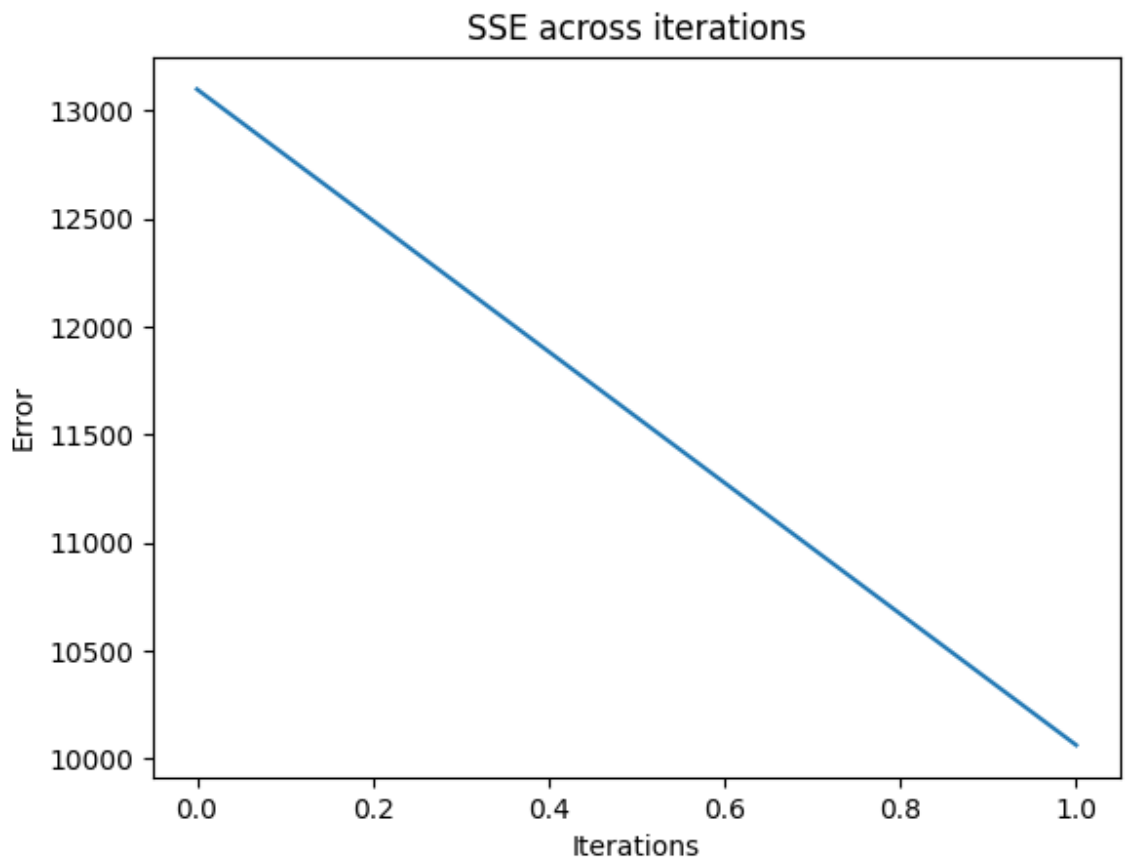
The lab is about the Levenberg–Marquardt (LM) Algorithm which is used to solve nonlinear least squares problems. This curve-fitting method is a combination of two other methods: the gradient descent and the Gauss-Newton.

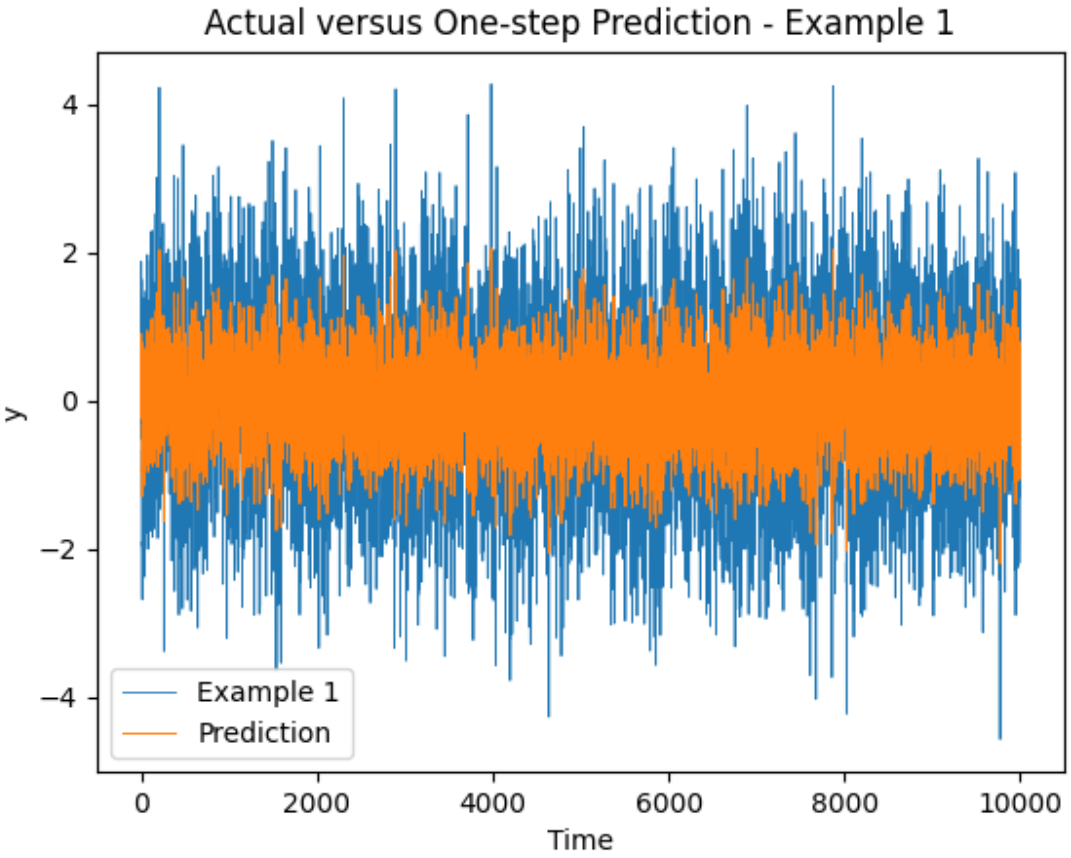
Introduction:

This experiment was performed to increase understanding of the application of the LM algorithm.

Answers to Lab Questions:

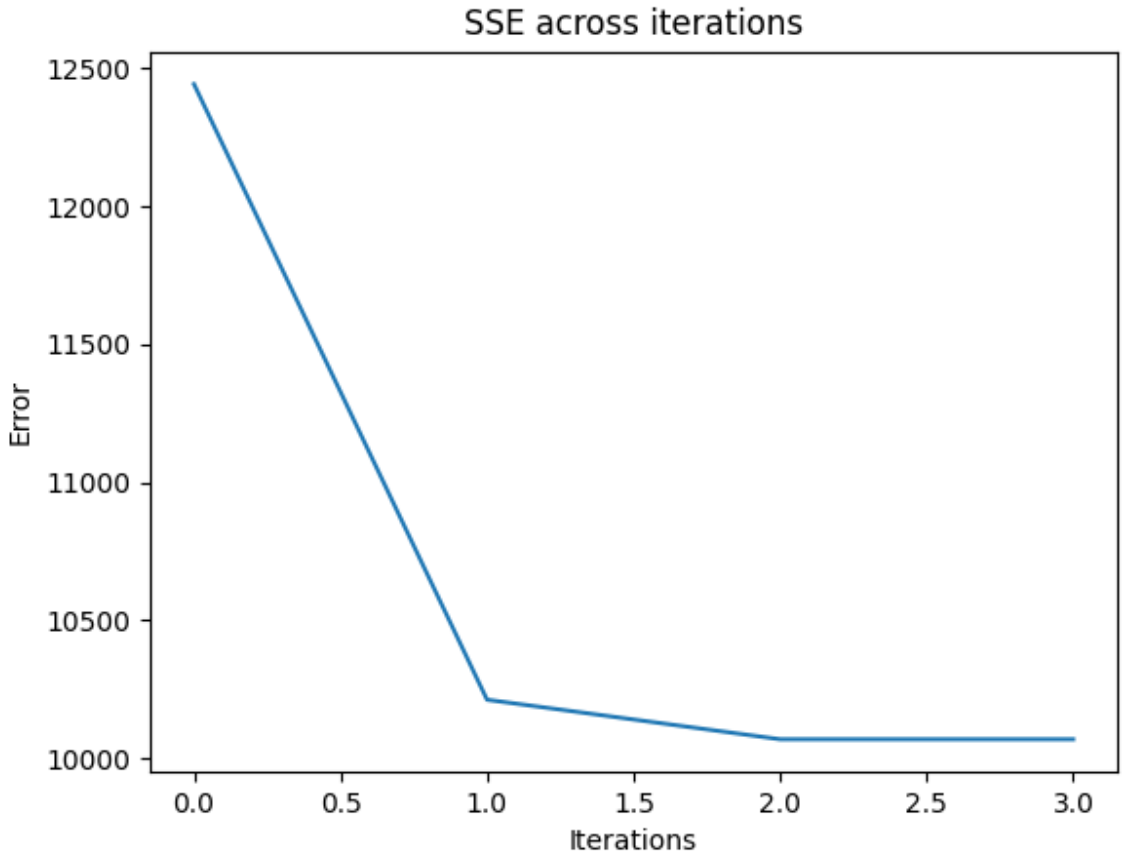
Example 1: $y(t) - 0.5y(t-1) = e(t)$ ARMA (1,0)

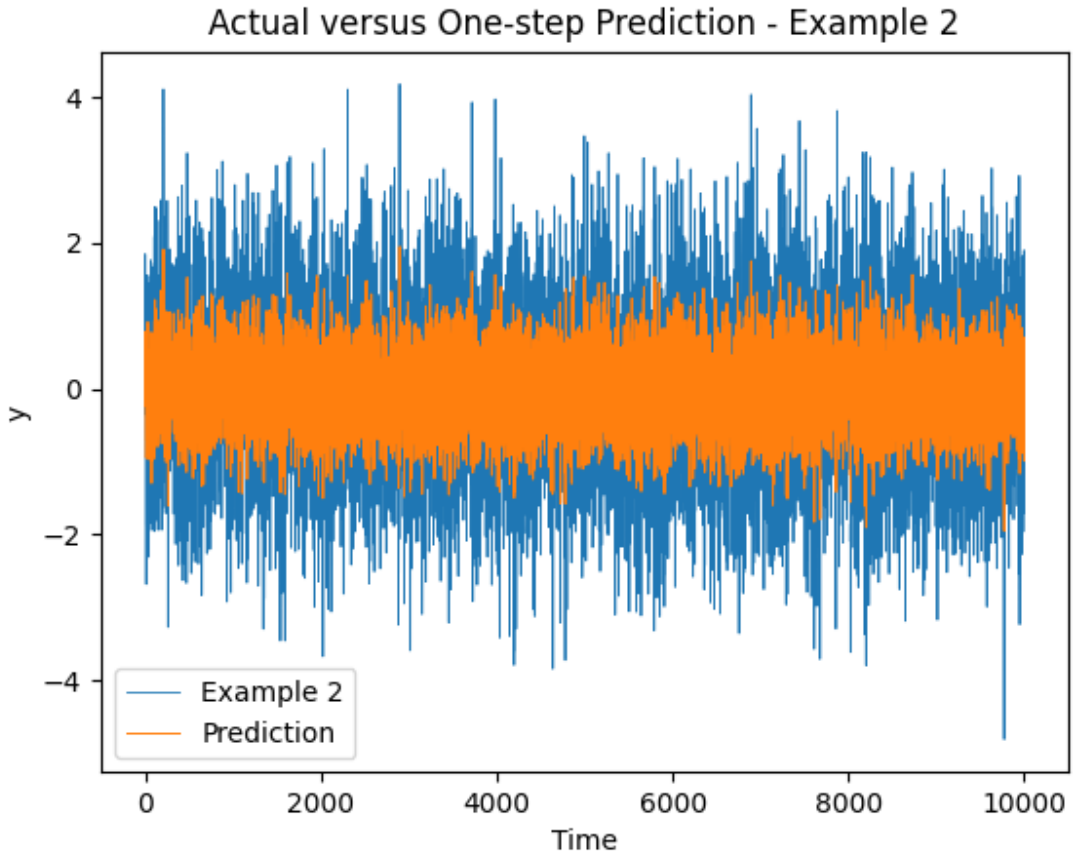





```
Convergence
Coefficients: [[-0.48140948]]
Confidence Interval of parameters
[-0.49894055] < theta_0 < [-0.4638784]
Covariance Matrix of estimated parameters:
 [[7.68346407e-05]]
Estimated variance of error:
 [[1.00648854]]
Roots of numerator: [0.]
Roots of denominator: [0.48140948]
None
Q value: 19.77480391264812
Q critical value 30.143527205646155
The residuals are white which means they are not correlated.
```

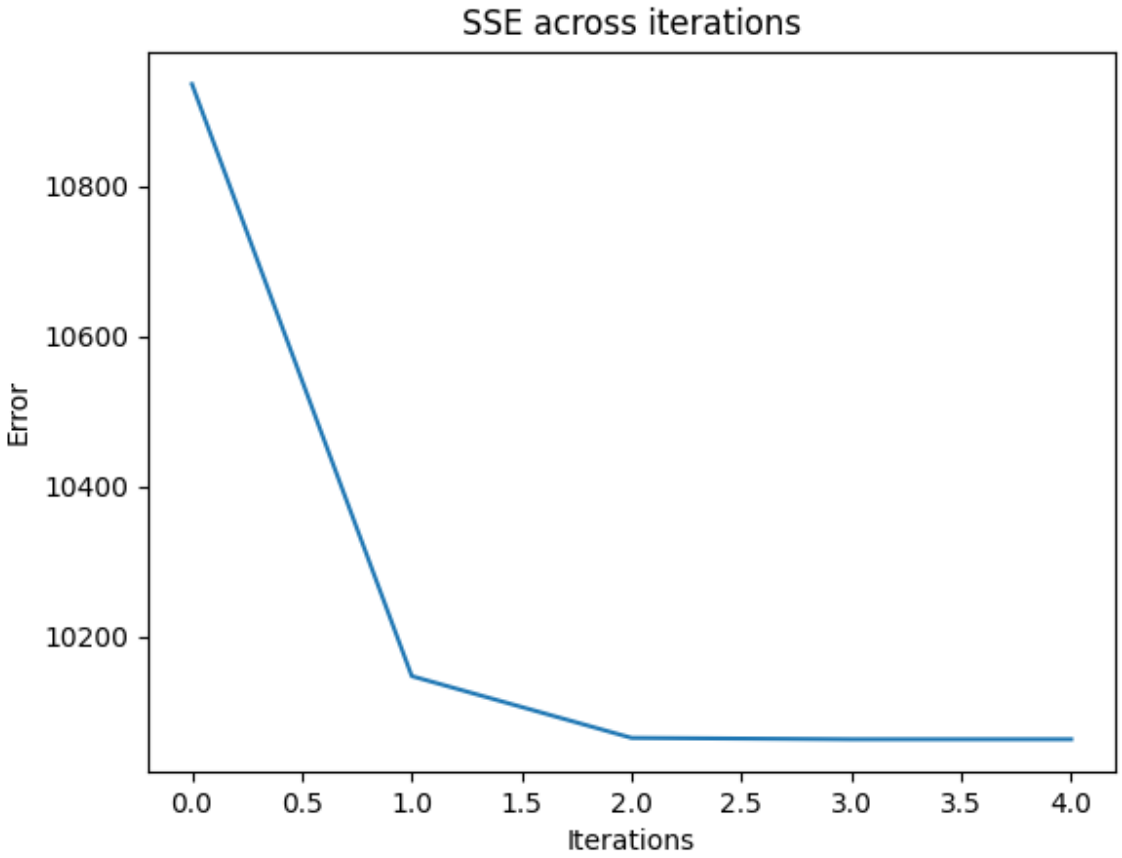
Example 2: ARMA (0,1): y(t) = e(t) + 0.5e(t-1)

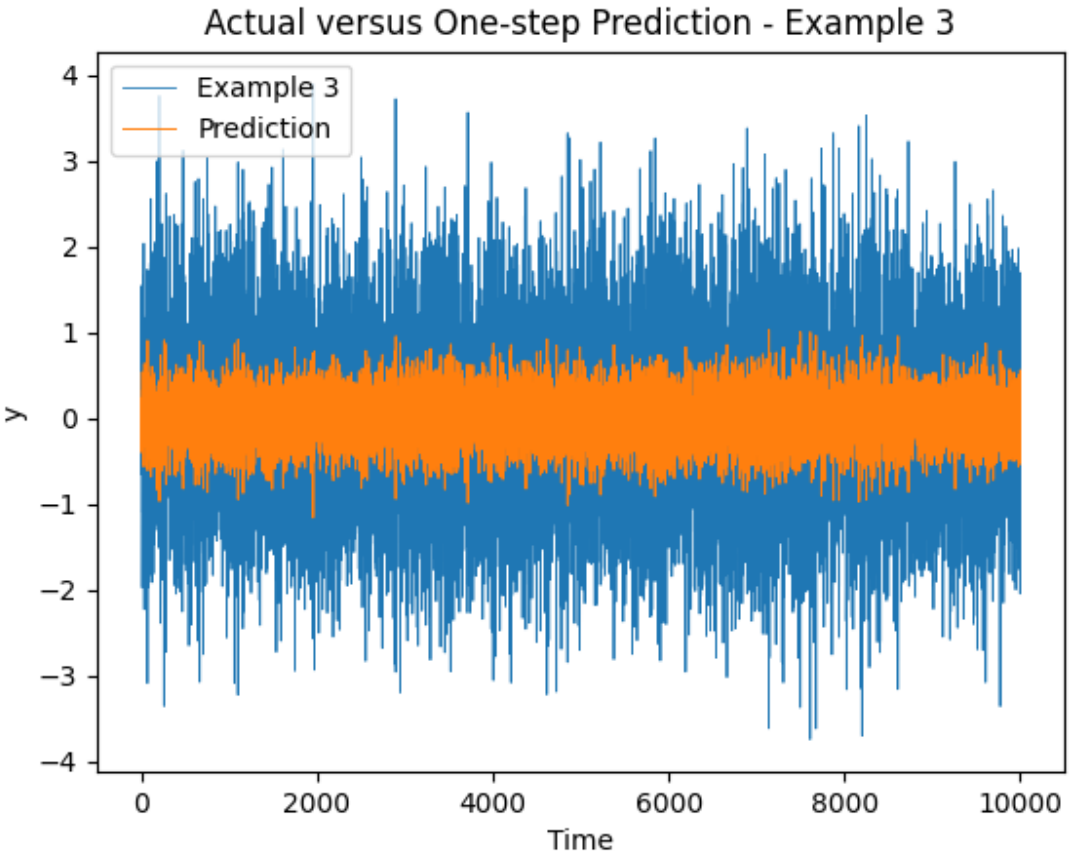

SSE across iterations
12500
12000
11500
11000
10500
10000
Error
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Iterations




```
Convergence
Coefficients: [[0.49628971]]
Confidence Interval of parameters
[0.47892397] < theta_0 < [0.51365544]
Covariance Matrix of estimated parameters:
 [[7.53922027e-05]]
Estimated variance of error:
 [[1.00692372]]
Roots of numerator: [-0.49628971]
Roots of denominator: [0.]
None
Q value: 24.468435398769277
Q critical value 30.143527205646155
The residuals are white which means they are not correlated.
```

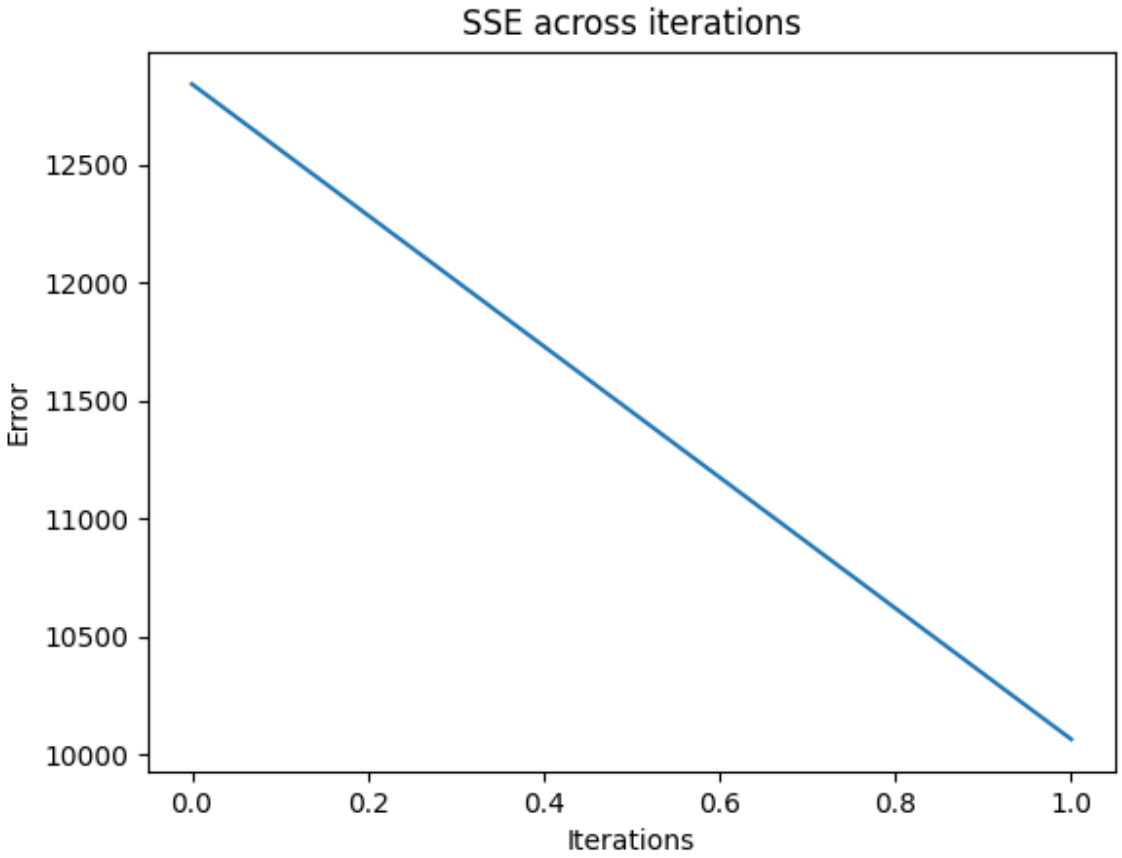
Example 3: ARMA (1,1): y(t) + 0.5y(t-1) = e(t) + 0.5e(t-1)

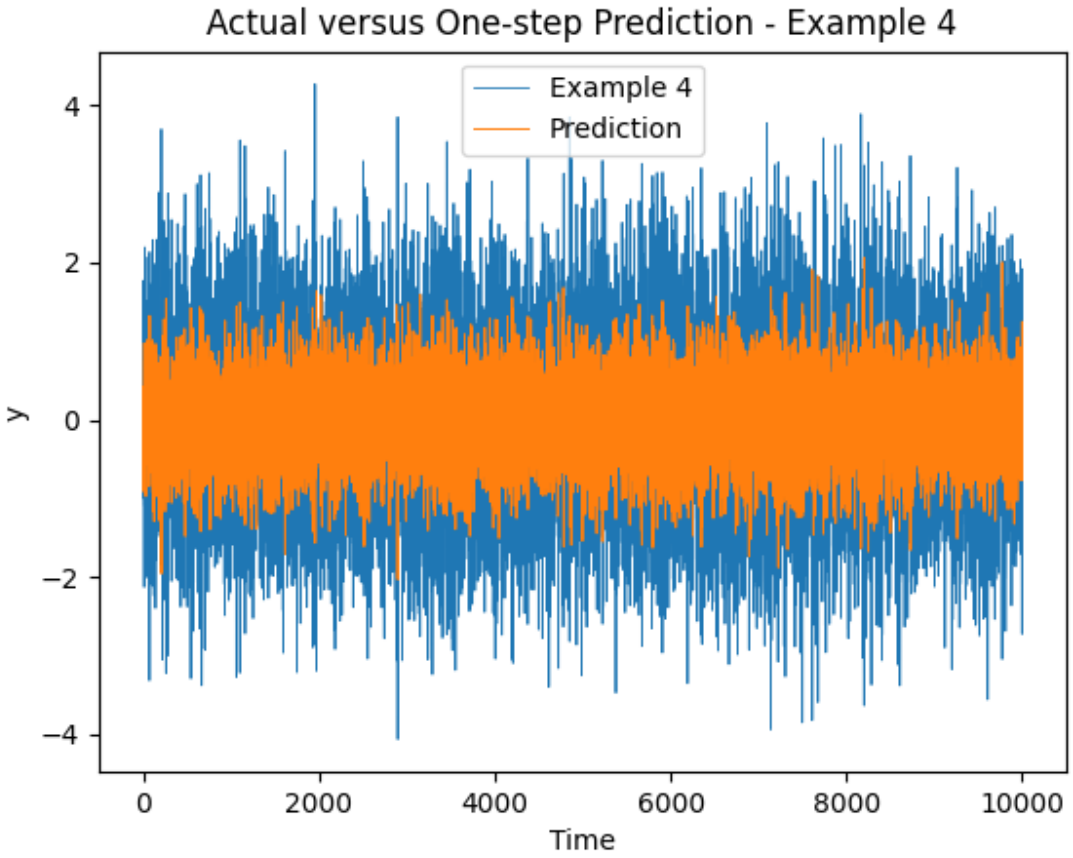

SSE across iterations
10800
10600
10400
10200
Error
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Iterations




```
Convergence
Coefficients: [[0.44351742]
 [0.17982477]]
Confidence Interval of parameters
[0.38095141] < theta_0 < [0.50608344]
[0.11115606] < theta_1 < [0.24849347]
Covariance Matrix of estimated parameters:
 [[0.00097863 0.00102904]
 [0.00102904 0.00117885]]
Estimated variance of error:
 [[1.00660456]]
Roots of numerator: [-0.17982477]
Roots of denominator: [-0.44351742]
None
Q value: 20.2237556750996
Q critical value 28.869299430392637
The residuals are white which means they are not correlated.
```

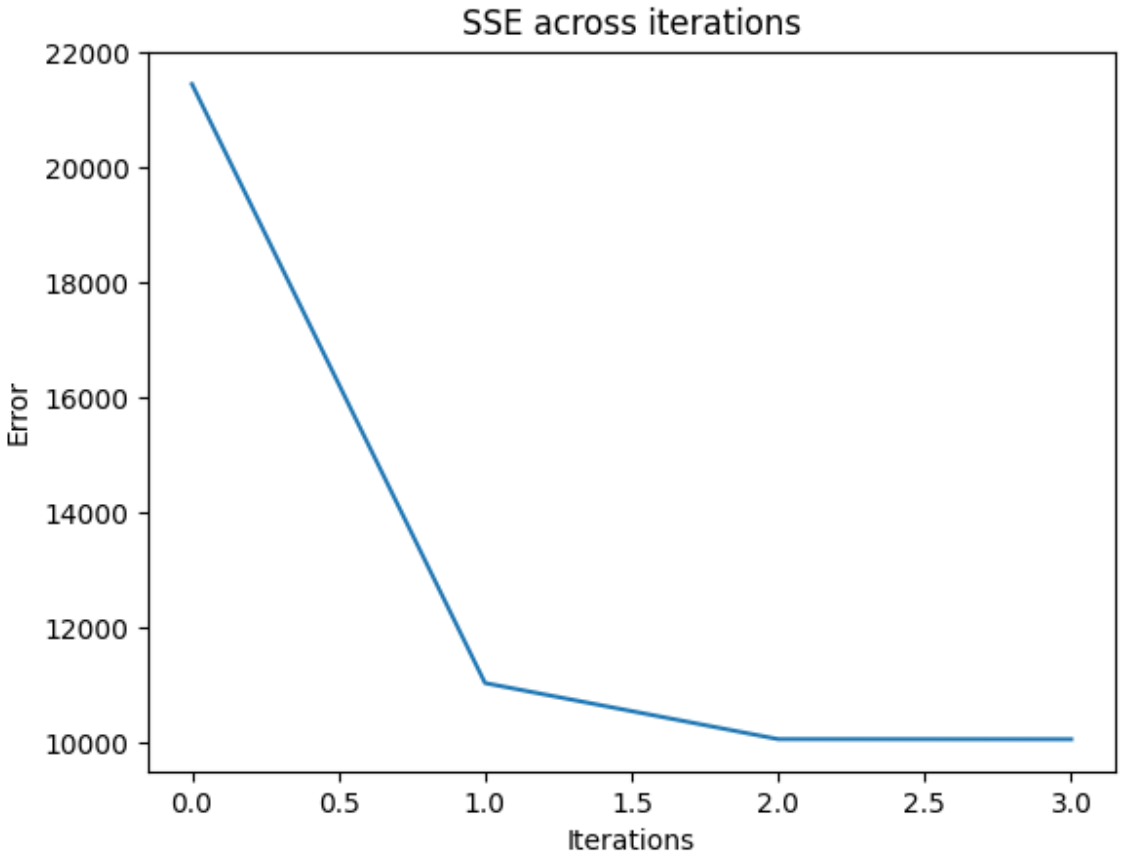
Example 4: ARMA (2,0): y(t) + 0.5y(t-1) + 0.2y(t-2) = e(t)

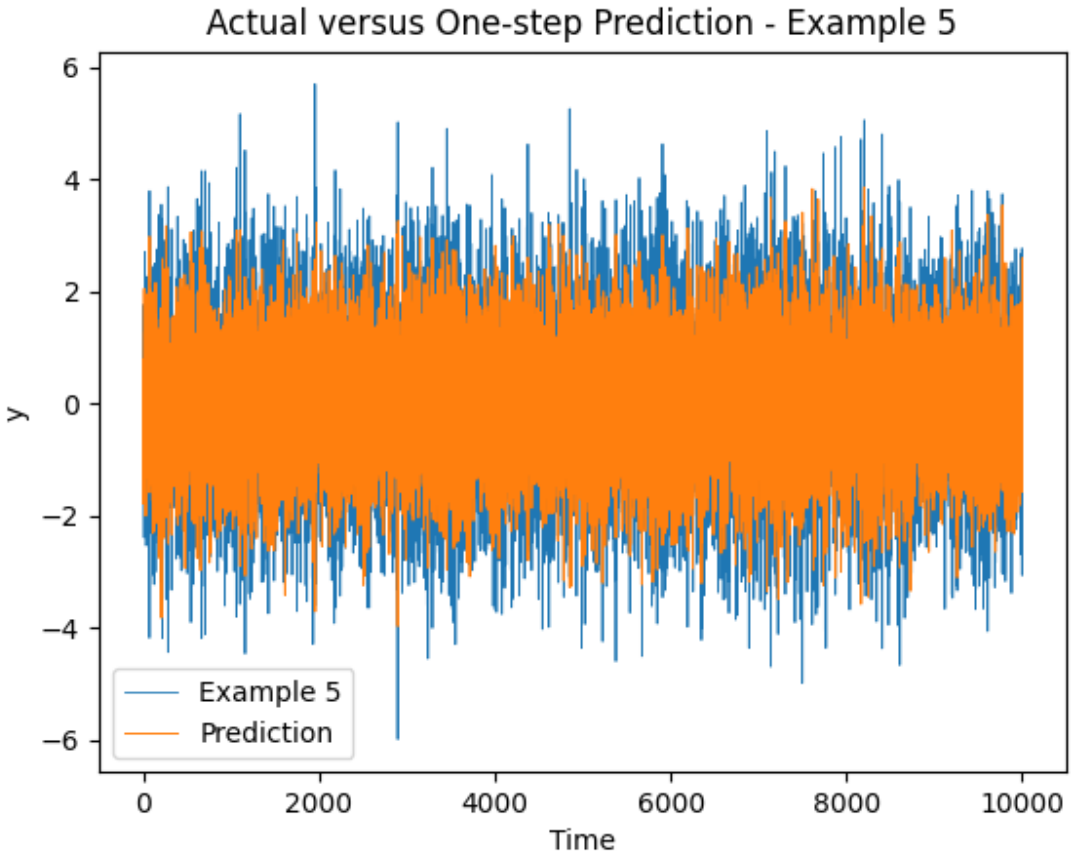

SSE across iterations
12500
12000
11500
11000
10500
10000
Error
0.0
0.2
0.4
0.6
0.8
1.0
Iterations




```
Convergence
Coefficients: [[0.51255554]
 [0.21733494]]
Confidence Interval of parameters
[0.4930314] < theta_0 < [0.53207968]
[0.19781034] < theta_1 < [0.23685955]
Covariance Matrix of estimated parameters:
 [[9.52980063e-05 4.01275824e-05]
 [4.01275824e-05 9.53025260e-05]]
Estimated variance of error:
 [[1.00668906]]
Roots of numerator: [0. 0.]
Roots of denominator: [-0.25627777+0.38943119j -0.25627777-0.38943119j]
None
Q value: 21.024127078563957
Q critical value 28.869299430392637
The residuals are white (not correlated).
```

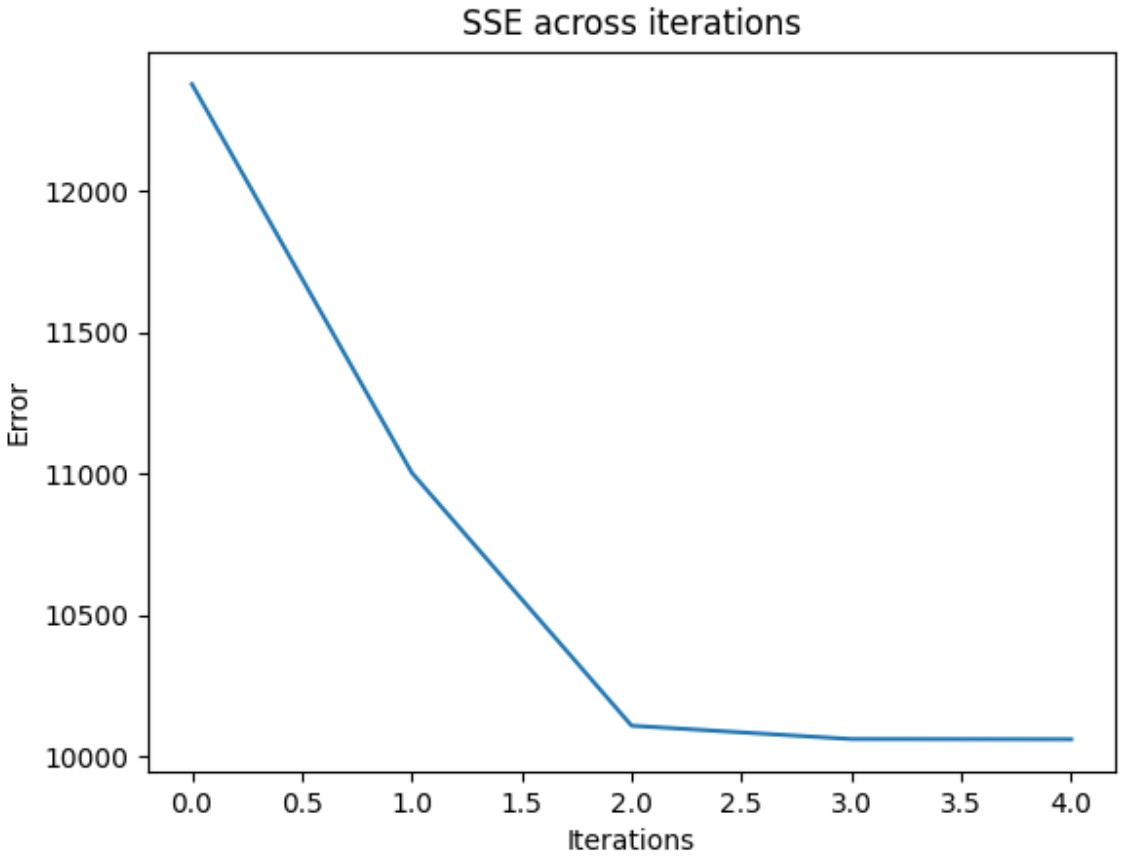
Example 5: ARMA (2,1): $y(t) + 0.5y(t-1) + 0.2y(t-2) = e(t) - 0.5e(t-1)$

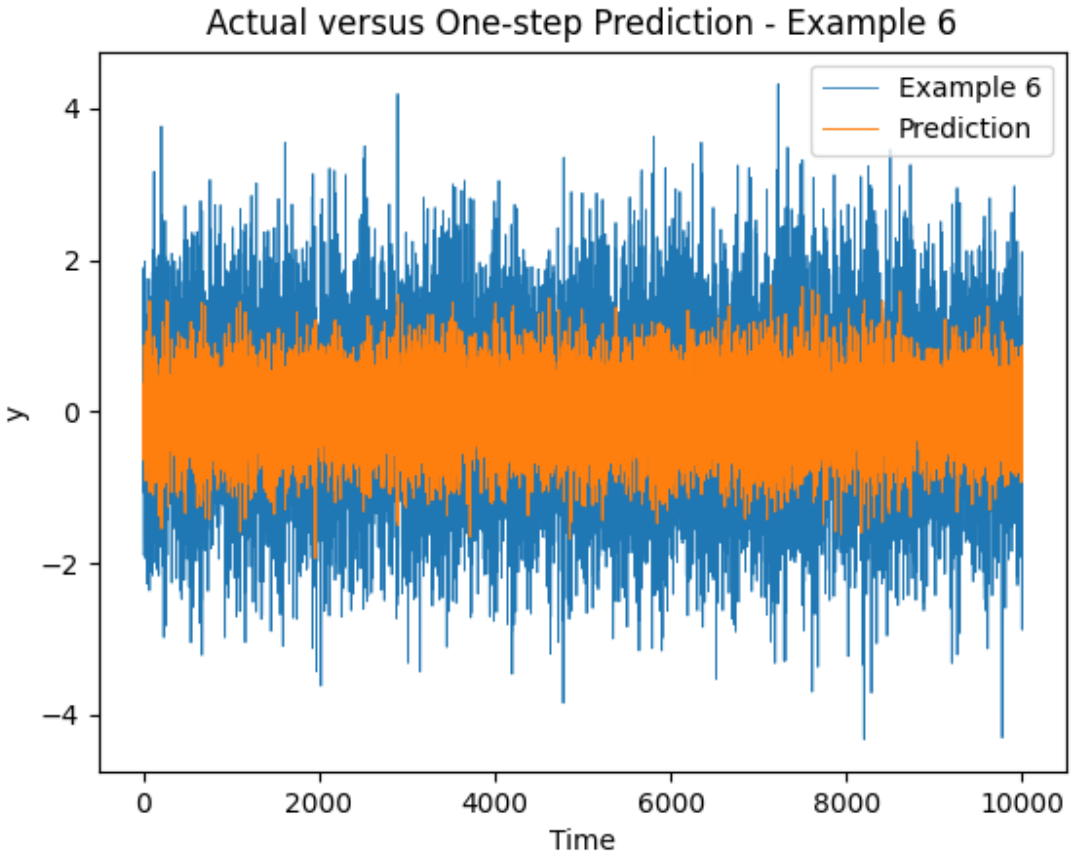

SSE across iterations
22000
20000
18000
16000
14000
12000
10000
Error
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Iterations


Actual versus One-step Prediction - Example 5
6
4
2
0
−2
−4
−6
y
Example 5
Prediction
0
2000
4000
6000
8000
10000
Time

```
Convergence
Coefficients: [[ 0.49416078]
 [ 0.20333726]
 [-0.52099308]]
Confidence Interval of parameters
[0.46068864] < theta_0 < [0.52763293]
[0.17474717] < theta_1 < [0.23192734]
[-0.55161884] < theta_2 < [-0.49036732]
Covariance Matrix of estimated parameters:
 [[0.0002801  0.00018075 0.00020783]
 [0.00018075 0.00020435 0.00015947]
 [0.00020783 0.00015947 0.00023448]]
Estimated variance of error:
 [[1.0066109]]
Roots of numerator: [0.52099308 0.        ]
Roots of denominator: [-0.24708039+0.37721153j -0.24708039-0.37721153j]
None
Q value: 19.295266357077065
Q critical value 27.587111638275335
The residuals are white (not correlated).
```

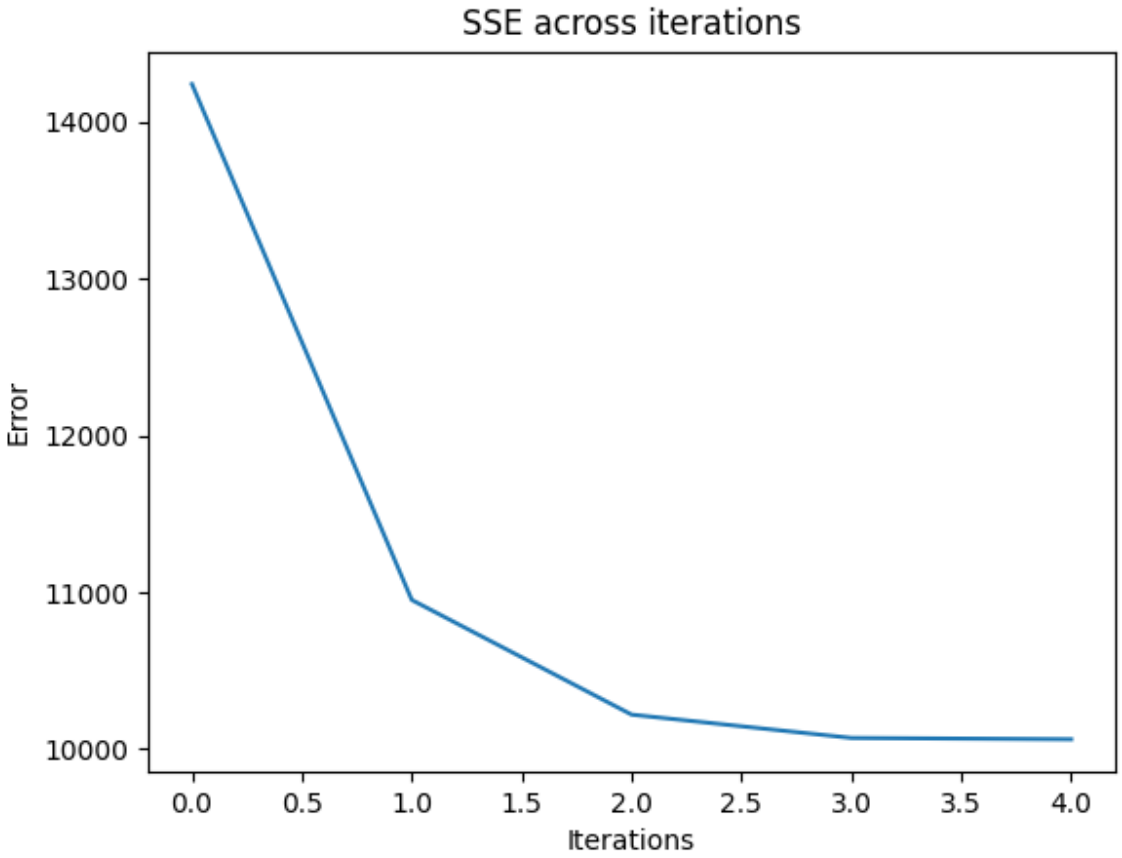
Example 6: ARMA (1,2): y(t) + 0.5y(t-1) = e(t) + 0.5e(t-1) - 0.4e(t-2)

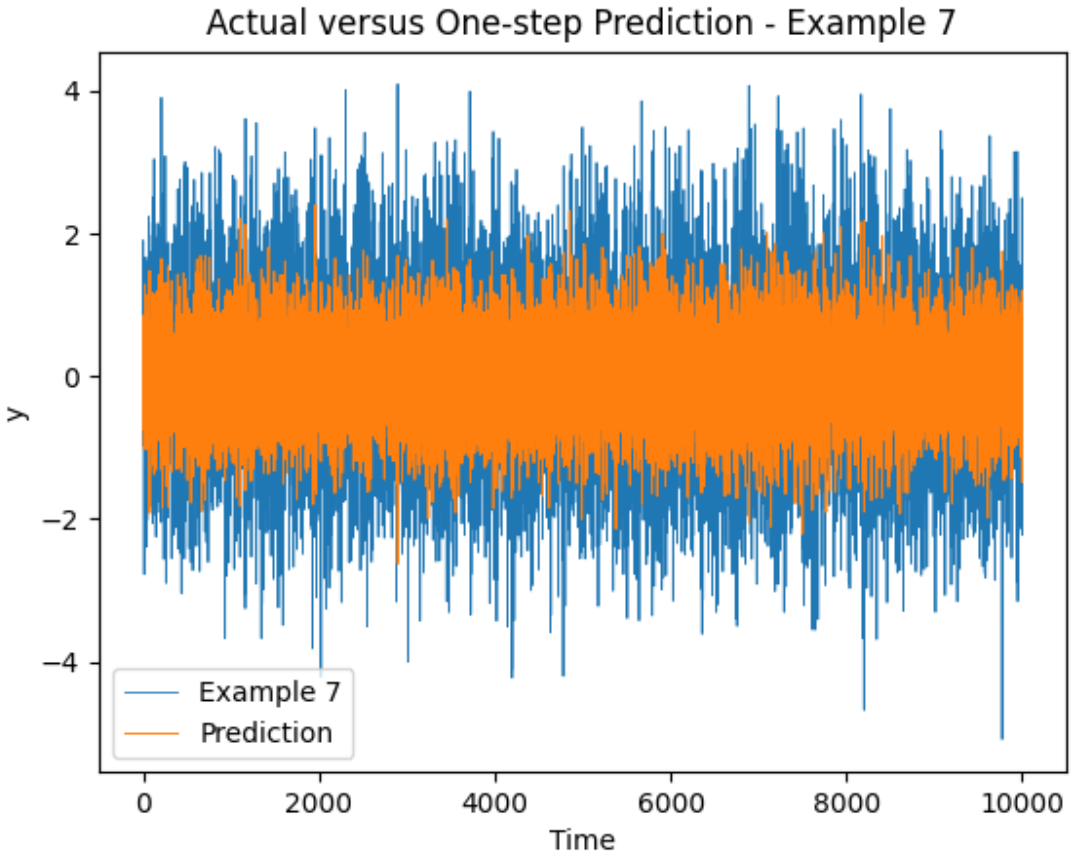

SSE across iterations
Error
12000
11500
11000
10500
10000
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Iterations


Actual versus One-step Prediction - Example 6
Example 6
Prediction
y
Time
4
2
0
−2
−4
0
2000
4000
6000
8000
10000

```
Convergence
Coefficients: [[ 0.489296  ]
 [ 0.47481313]
 [-0.42533876]]
Confidence Interval of parameters
[0.4614788] < theta_0 < [0.5171132]
[0.44753675] < theta_1 < [0.50208951]
[-0.44678756] < theta_2 < [-0.40388996]
Covariance Matrix of estimated parameters:
 [[1.93449164e-04 1.41874679e-04 7.99625042e-05]
 [1.41874679e-04 1.86000265e-04 1.26238008e-04]
 [7.99625042e-05 1.26238008e-04 1.15012729e-04]]
Estimated variance of error:
 [[1.0065695]]
Roots of numerator: [-0.93145313  0.45664   ]
Roots of denominator: [-0.489296  0.      ]
None
Q value: 18.836218870030486
Q critical value 27.587111638275335
The residuals are white which means they are not correlated).
```

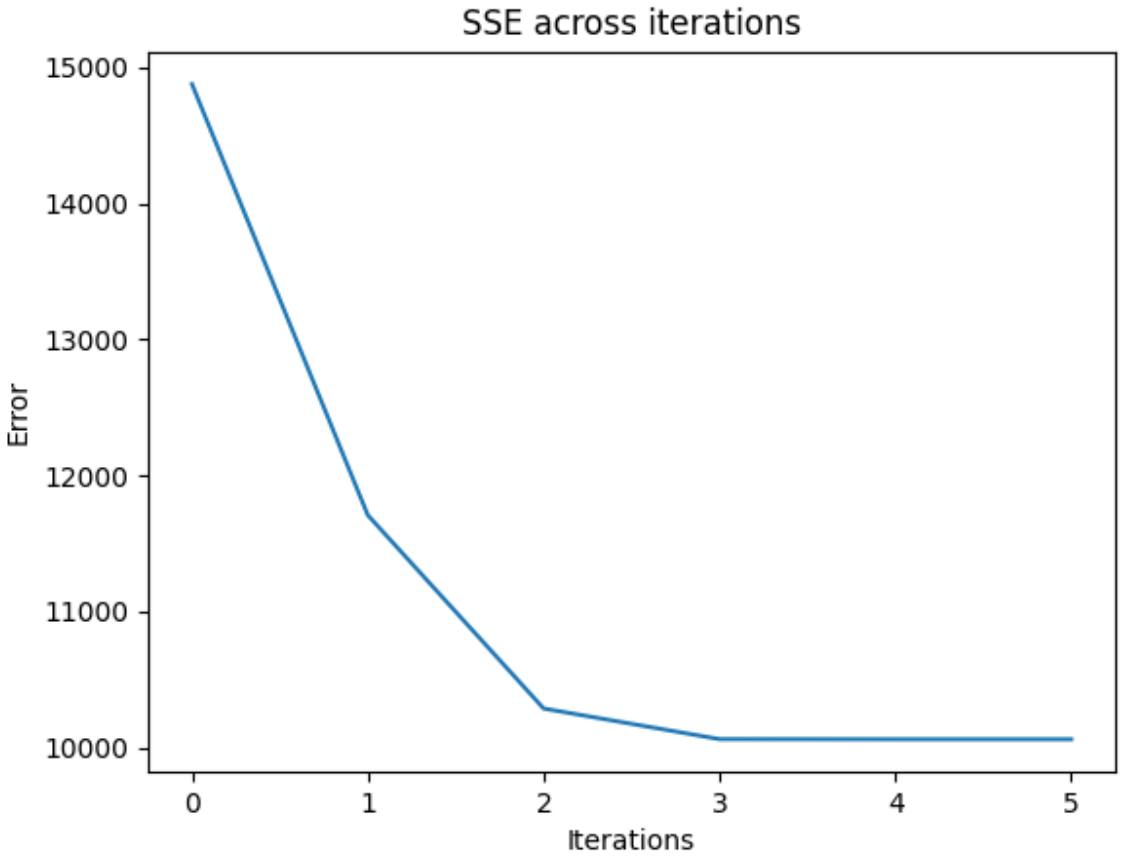
Example 7: ARMA (0,2): y(t) = e(t) + 0.5e(t-1) - 0.4e(t-2)

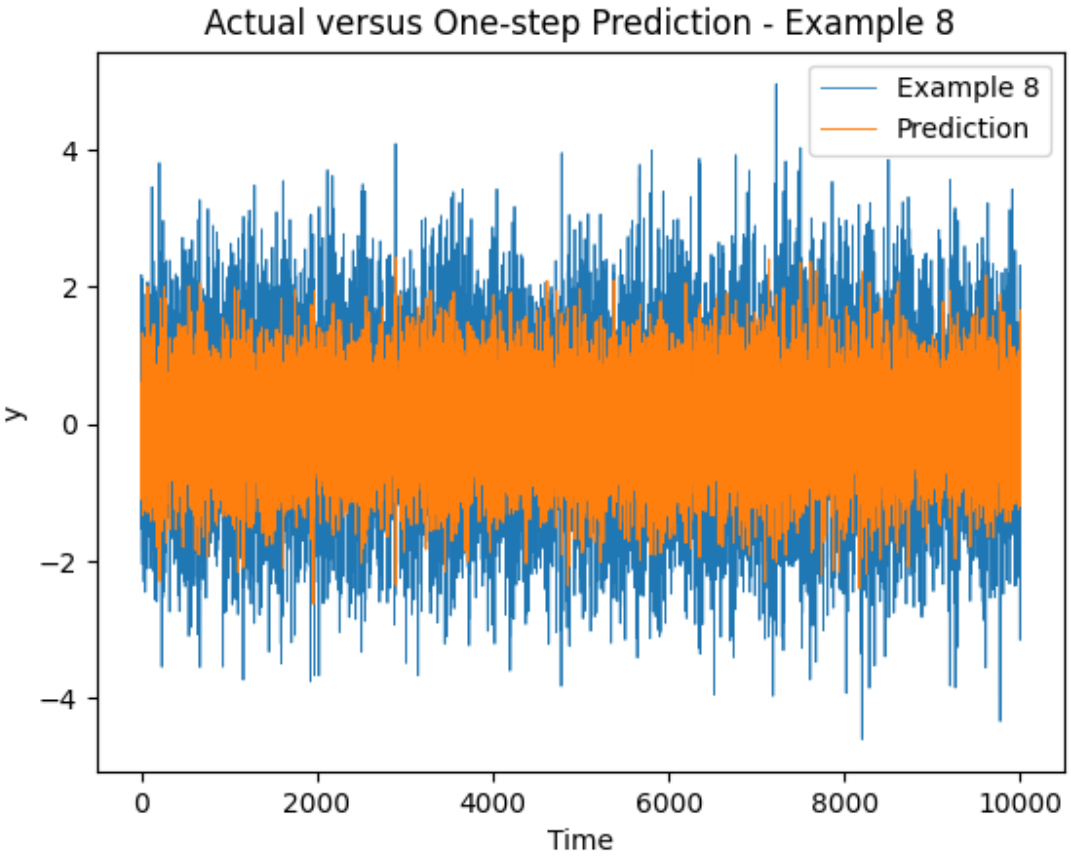

SSE across iterations
14000
13000
12000
11000
10000
Error
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Iterations




```
Convergence
Coefficients: [[ 0.48262171]
 [-0.42066298]]
Confidence Interval of parameters
[0.46447275] < theta_0 < [0.50077067]
[-0.43881312] < theta_1 < [-0.40251284]
Covariance Matrix of estimated parameters:
 [[8.23461854e-05 6.85251679e-05]
 [6.85251679e-05 8.23569012e-05]]
Estimated variance of error:
 [[1.00653193]]
Roots of numerator: [-0.93333246  0.45071076]
Roots of denominator: [0. 0.]
None
Q value: 19.11577542099665
Q critical value 28.869299430392637
The residuals are white (not correlated).
```

Example 8: ARMA (2,2): $y(t)+0.5y(t-1) +0.2y(t-2) = e(t)+0.5e(t-1) - 0.4e(t-2)$


SSE across iterations
15000
14000
13000
12000
11000
10000
Error
0
1
2
3
4
5
Iterations


Actual versus One-step Prediction - Example 8
Example 8
Prediction
y
Time
4
2
0
−2
−4
0
2000
4000
6000
8000
10000

```
Convergence
Coefficients: [[ 0.49164012]
 [ 0.19918853]
 [ 0.47671789]
 [-0.4249688 ]]
Confidence Interval of parameters
[0.4539713] < theta_0 < [0.52930894]
[0.1665096] < theta_1 < [0.23186746]
[0.44100448] < theta_2 < [0.5124313]
[-0.46025624] < theta_3 < [-0.38968135]
Covariance Matrix of estimated parameters:
 [[0.00035473 0.00022875 0.00028714 0.00026847]
 [0.00022875 0.00026698 0.00020827 0.000229  ]
 [0.00028714 0.00020827 0.00031886 0.00029607]
 [0.00026847 0.000229   0.00029607 0.0003113 ]]
Estimated variance of error:
 [[1.00669537]]
Roots of numerator: [-0.93246541  0.45574752]
Roots of denominator: [-0.24582006+0.37250641j -0.24582006-0.37250641j]
None
Q value: 18.94966802795232
Q critical value 26.296227604864242
The residuals are white (not correlated).
```

8. It seems that higher order ARMA models have less static ACF and PACF graphs.

Conclusion:

LM algorithms can be used to estimate the coefficient of ARMA models.

Appendix:

```
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
import seaborn as sns
import warnings
import pandas as pd
import Toolbox
from scipy import signal, linalg
from sklearn.model_selection import train_test_split
import scipy
```

```
warnings.filterwarnings('ignore')


def plot_prediction(train, prediction, label):
    plt.plot(train, label=label, lw=0.6)
    plt.plot(prediction, label="Prediction", lw=0.7)
    plt.title("Actual versus One-step Prediction - " + label)
    plt.xlabel("Time")
    plt.ylabel("y")
    plt.legend()
    plt.show()




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

print("-------------------- Example 1 ----------------------")

y = Toolbox.generate_arma_no_questions(10000, 0, 1, 1, 0, [1, -0.5], [1, 0])

t = Toolbox.levenberg_marquardt(y, 1, 0)


def onestep_prediction_example1(y, t):
    # y(t) - 0.5*y(t - 1) = e(t)
    # e(t) = y(t) - yhat(t-1)
    # y(t+1) = 0.5y(t)
    p = np.zeros(shape=y.shape)
    for i in range(1, len(y)):
        p[i] = -t[0] * y[i - 1]
    return p


prediction = onestep_prediction_example1(y, t)
plot_prediction(y, prediction, "Example 1")
residual = y - prediction
residual = residual[1:]

Q = Toolbox.box_pierce_test(y, residual, 20)  # test for white noise - box
pierce test
df = 20 - len(t)  # DOF = h - na - nb
Qc = scipy.stats.chi2.isf(0.05, df)  # chisq test for whiteness
print("Q value:", Q, "\nQ critical value", Qc)
if Q < Qc:
    print("The residuals are white which means they are not correlated.")
else:
    print("The residuals are not white which means they are correlated.")



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

print("-------------------- Example 2 ----------------------")
```

```
y = Toolbox.generate_arma_no_questions(10000, 0, 1, 0, 1, [1, 0], [1, 0.5])

t = Toolbox.levenberg_marquardt(y, 0, 1)


def onestep_prediction_example2(y, t):
    # y(t) = e(t) + 0.5e(t-1)
    # e(t) = y(t) - yhat(t-1)
    # y(t+1) = 0.5y(t)-0.5(pred[t])
    p = np.empty(shape=y.shape)
    for i in range(1, len(y)):
        if i == 1:
            p[i] = t[0] * y[i - 1]
        else:
            p[i] = t[0] * y[i - 1] - t[0] * p[i - 1]
    return p


prediction = onestep_prediction_example2(y, t)
plot_prediction(y, prediction, "Example 2")
residual = y - prediction
residual = residual[1:]

Q = Toolbox.box_pierce_test(y, residual, 20)  # test for white noise - box
pierce test
df = 20 - len(t)  # DOF = h - na - nb
Qc = scipy.stats.chi2.isf(0.05, df)  # chisq test for whiteness
print("Q value:", Q, "\nQ critical value", Qc)
if Q < Qc:
    print("The residuals are white which means they are not correlated.")
else:
    print("The residuals are not white which means they are correlated.")



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

print("--------------------- Example 3 ----------------------")

y = Toolbox.generate_arma_no_questions(10000, 0, 1, 1, 1, [1, 0.5], [1,
0.25])

t = Toolbox.levenberg_marquardt(y, 1, 1)


def onestep_prediction_example3(y, t):
    # ARMA (1,1): y(t) + 0.5y(t-1) = e(t) + 0.25e(t-1)
    # e(t) = y(t) - yhat(t-1)
    # y(t+1) = 0.5y(t)-0.5(pred[t])
    p = np.empty(shape=y.shape)
    for i in range(1, len(y)):
        if i == 1:
            p[i] = -t[0] * y[i - 1] + t[1] * y[i - 1]
        else:
            p[i] = -t[0] * y[i - 1] + t[1] * y[i - 1] - t[1] * p[i - 1]
    return p
```

```
prediction = onestep_prediction_example3(y, t)
plot_prediction(y, prediction, "Example 3")
residual = y - prediction
residual = residual[1:]

Q = Toolbox.box_pierce_test(y, residual, 20)  # test for white noise - box
pierce test
df = 20 - len(t)  # DOF = h - na - nb
Qc = scipy.stats.chi2.isf(0.05, df)  # chisq test for whiteness
print("Q value:", Q, "\nQ critical value", Qc)
if Q < Qc:
    print("The residuals are white which means they are not correlated.")
else:
    print("The residuals are not which means they are white correlated.")



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

print("-------------------- Example 4 ----------------------")

y = Toolbox.generate_arma_no_questions(10000, 0, 1, 2, 0, [1, 0.5, 0.2], [1])

t = Toolbox.levenberg_marquardt(y, 2, 0)


def onestep_prediction_example4(y, t):
    # ARMA (2,0): y(t) + 0.5y(t-1) + 0.2y(t-2) = e(t)
    # e(t) = y(t) - yhat(t-1)
    # y(t+1) = 0.5y(t)-0.5(pred[t])
    p = np.empty(shape=y.shape)
    for i in range(1, len(y)):
        if i == 1:
            p[i] = -t[0] * y[i - 1]
        else:
            p[i] = -t[0] * y[i - 1] - t[1] * y[i - 2]
    return p


prediction = onestep_prediction_example4(y, t)
plot_prediction(y, prediction, "Example 4")
residual = y - prediction
residual = residual[1:]

Q = Toolbox.box_pierce_test(y, residual, 20)  # test for white noise - box
pierce test
df = 20 - len(t)  # DOF = h - na - nb
Qc = scipy.stats.chi2.isf(0.05, df)  # chisq test for whiteness
print("Q value:", Q, "\nQ critical value", Qc)
if Q < Qc:
    print("The residuals are white (not correlated).")
else:
    print("The residuals are not white (correlated).")
```

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

print("-------------------- Example 5 ----------------------")

y = Toolbox.generate_arma_no_questions(10000, 0, 1, 2, 1, [1, 0.5, 0.2], [1,
-0.5])

t = Toolbox.levenberg_marquardt(y, 2, 1)


def onestep_prediction_example5(y, t):
    # Example 5: ARMA (2,1): y(t) + 0.5y(t-1) + 0.2y(t-2) = e(t) - 0.5e(t-1)
    # e(t) = y(t) - yhat(t-1)
    # y(t+1) = 0.5y(t)-0.5(pred[t])
    p = np.empty(shape=y.shape)
    for i in range(1, len(y)):
        if i == 1:
            p[i] = -t[0] * y[i - 1] - t[2] * y[i - 1]
        else:
            p[i] = -t[0] * y[i - 1] - t[1] * y[i - 2] + t[2] * y[i - 1] -
t[2] * p[i - 1]
    return p


prediction = onestep_prediction_example5(y, t)
plot_prediction(y, prediction, "Example 5")
residual = y - prediction
residual = residual[1:]

Q = Toolbox.box_pierce_test(y, residual, 20)  # test for white noise - box
pierce test
df = 20 - len(t)  # DOF = h - na - nb
Qc = scipy.stats.chi2.isf(0.05, df)  # chisq test for whiteness
print("Q value:", Q, "\nQ critical value", Qc)
if Q < Qc:
    print("The residuals are white (not correlated).")
else:
    print("The residuals are not white (correlated).")



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

print("-------------------- Example 6 ----------------------")

y = Toolbox.generate_arma_no_questions(10000, 0, 1, 1, 2, [1, 0.5], [1, 0.5,
-0.4])

t = Toolbox.levenberg_marquardt(y, 1, 2)


def onestep_prediction_example6(y, t):
    # Example 6: ARMA (1,2): y(t) + 0.5y(t-1) = e(t) + 0.5e(t-1) - 0.4e(t-2)
    # e(t) = y(t) - yhat(t-1)
    # y(t+1) = 0.5y(t)-0.5(pred[t])
    p = np.empty(shape=y.shape)
```

```
    for i in range(1, len(y)):
        if i == 1:
            p[i] = -t[0] * y[i - 1] + t[1] * y[i - 1]
        elif i == 2:
            p[i] = -t[0] * y[i - 1] + t[1] * y[i - 1] - t[1] * p[i - 1] +
t[2] * y[i - 2]
        else:
            p[i] = -t[0] * y[i - 1] + t[1] * y[i - 1] - t[1] * p[i - 1] +
t[2] * y[i - 2] - \
                        t[2] * p[i - 2]
    return p


prediction = onestep_prediction_example6(y, t)
plot_prediction(y, prediction, "Example 6")
residual = y - prediction
residual = residual[1:]

Q = Toolbox.box_pierce_test(y, residual, 20)  # test for white noise - box
pierce test
df = 20 - len(t)  # DOF = h - na - nb
Qc = scipy.stats.chi2.isf(0.05, df)  # chisq test for whiteness
print("Q value:", Q, "\nQ critical value", Qc)
if Q < Qc:
    print("The residuals are white which means they are not correlated).")
else:
    print("The residuals are not white which means they are correlated.")



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

print("-------------------- Example 7 ----------------------")

y = Toolbox.generate_arma_no_questions(10000, 0, 1, 0, 2, [1], [1, 0.5, -
0.4])

t = Toolbox.levenberg_marquardt(y, 0, 2)


def onestep_prediction_example7(y, t):
    # Example 7: ARMA (0,2): y(t) = e(t) + 0.5e(t-1) - 0.4e(t-2)
    # e(t) = y(t) - yhat(t-1)
    # y(t+1) = 0.5y(t)-0.5(pred[t])
    p = np.empty(shape=y.shape)
    for i in range(1, len(y)):
        if i == 1:
            p[i] = t[0] * y[i - 1]
        elif i == 2:
            p[i] = t[0] * y[i - 1] - t[0] * p[i - 1] + t[1] * y[i - 2]
        else:
            p[i] = t[0] * y[i - 1] - t[0] * p[i - 1] + t[1] * y[i - 2] - t[1]
* p[i - 2]
    return p


prediction = onestep_prediction_example7(y, t)
```

```
plot_prediction(y, prediction, "Example 7")
residual = y - prediction
residual = residual[1:]

Q = Toolbox.box_pierce_test(y, residual, 20)  # test for white noise - box
pierce test
df = 20 - len(t)  # DOF = h - na - nb
Qc = scipy.stats.chi2.isf(0.05, df)  # chisq test for whiteness
print("Q value:", Q, "\nQ critical value", Qc)
if Q < Qc:
    print("The residuals are white (not correlated).")
else:
    print("The residuals are not white (correlated).")



# Example 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("-------------------- Example 8 ----------------------")

y = Toolbox.generate_arma_no_questions(10000, 0, 1, 2, 2, [1, 0.5, 0.2], [1,
0.5, -0.4])

t = Toolbox.levenberg_marquardt(y, 2, 2)


def onestep_prediction_example8(y, t):
    # Example 8: ARMA (2,2): y(t)+0.5y(t-1) +0.2y(t-2) = e(t)+0.5e(t-1) -
0.4e(t-2)
    # e(t) = y(t) - yhat(t-1)

    p = np.empty(shape=y.shape)
    for i in range(1, len(y)):
        if i == 1:
            p[i] = -t[0] * y[i - 1] + t[2] * y[i - 1]
        elif i == 2:
            p[i] = -t[0] * y[i - 1] - t[1] * y[i - 2] + t[2] * (y[i - 1] -
p[i - 1]) + t[3] * y[i - 2]
        else:
            p[i] = -t[0] * y[i - 1] - t[1] * y[i - 2] \
                   + t[2] * (y[i - 1] - p[i - 1]) \
                   + t[3] * (y[i - 2] - p[i - 2])
    return p


prediction = onestep_prediction_example8(y, t)
plot_prediction(y, prediction, "Example 8")
residual = y - prediction
residual = residual[1:]

Q = Toolbox.box_pierce_test(y, residual, 20)  # test for white noise - box
pierce test
df = 20 - len(t)  # DOF = h - na - nb
Qc = scipy.stats.chi2.isf(0.05, df)  # chisq test for whiteness
print("Q value:", Q, "\nQ critical value", Qc)
if Q < Qc:
```

```
    print("The residuals are white (not correlated).")
else:
    print("The residuals are not white (correlated).")
```