**DATS 6313 – Time Series Analysis & Modeling**

Instructor: Reza Jafari

**Lab #8**

Bradley Reardon

4/13/2022

**Abstract:**

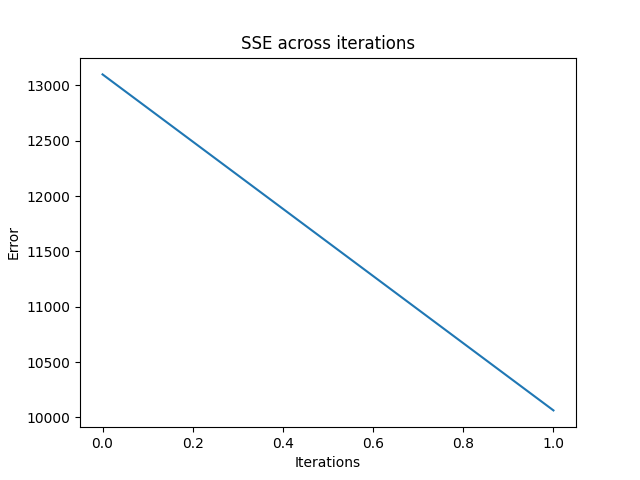
The lab is about the Levenberg–Marquardt (LM) Algorithm which is used to solve nonlinear least squares problems. This curve-fitting method is a combination of two other methods: the gradient descent and the Gauss-Newton.

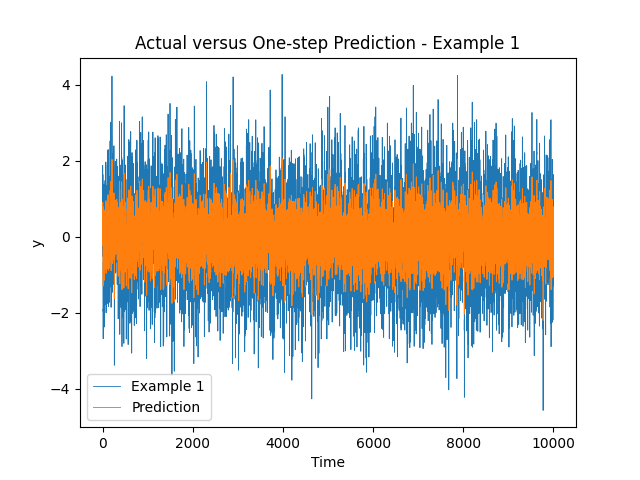
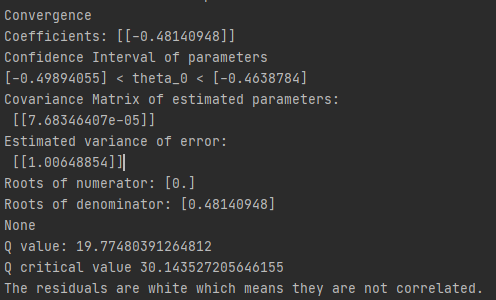
**Introduction:**

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

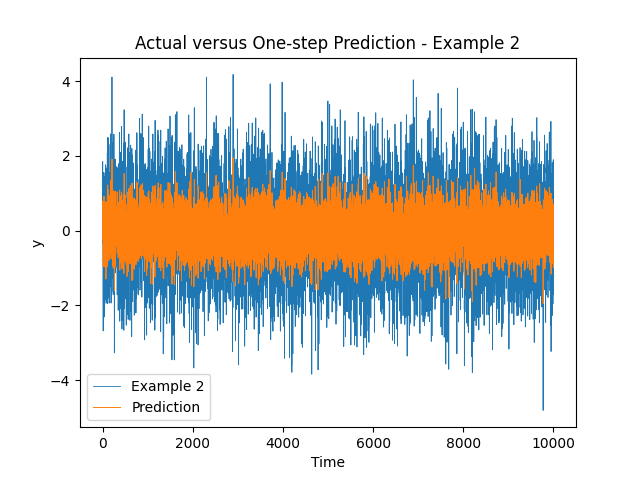
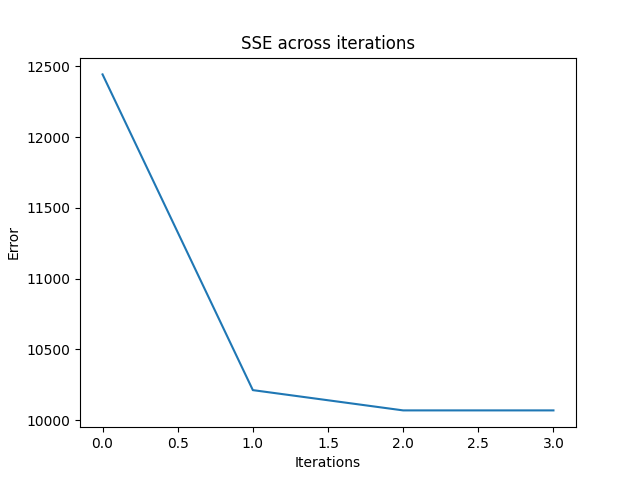
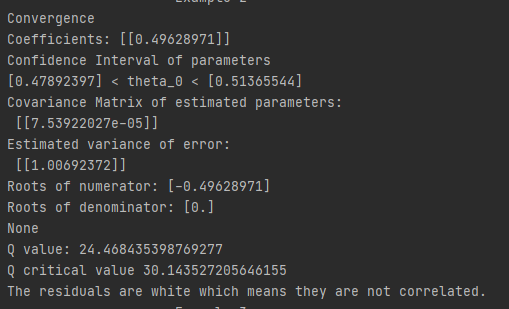
**Answers to Lab Questions:**

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

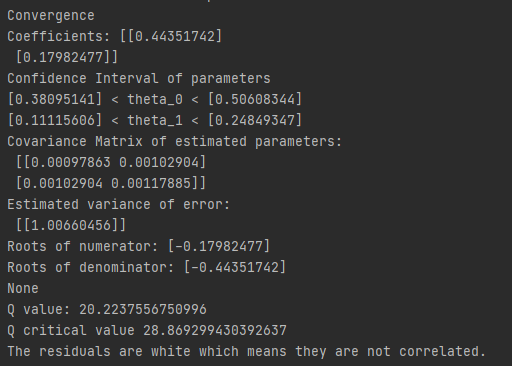
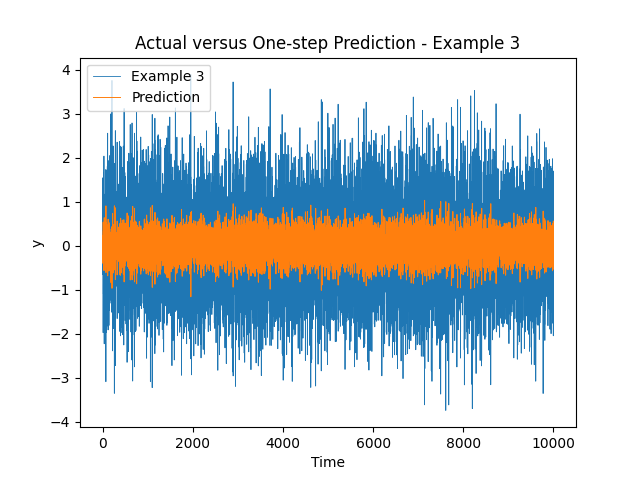
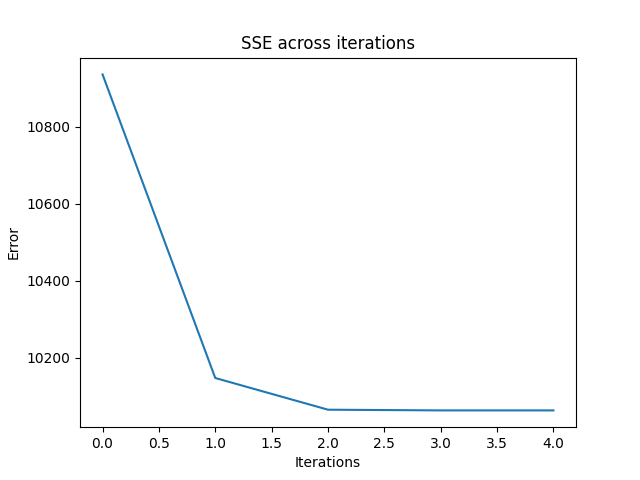
****

****

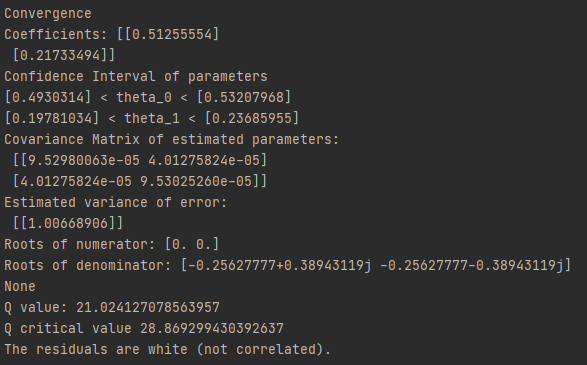
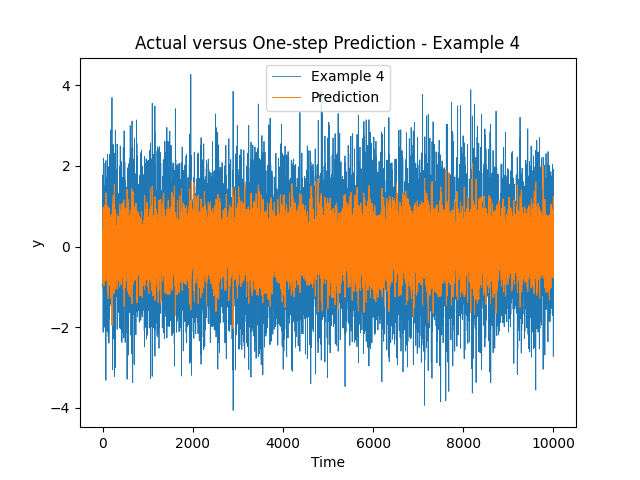
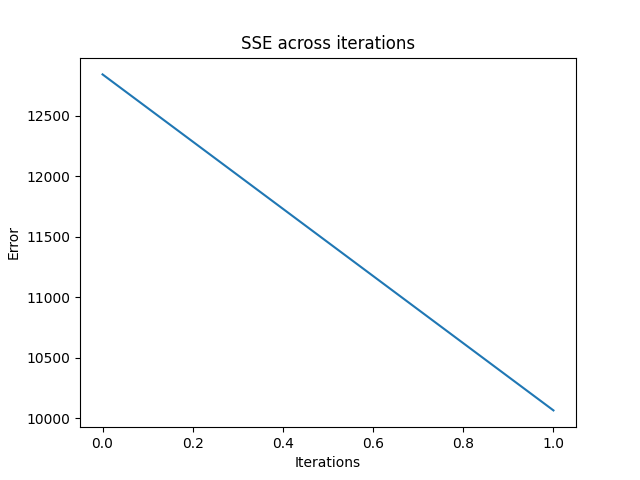
Example 2: ARMA (0,1): y(t) = e(t) + 0.5e(t-1)

****

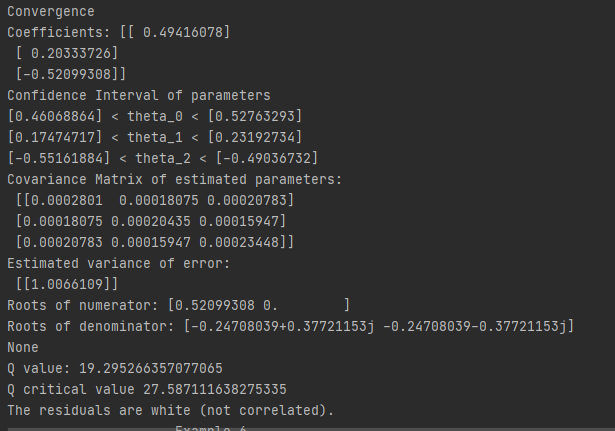
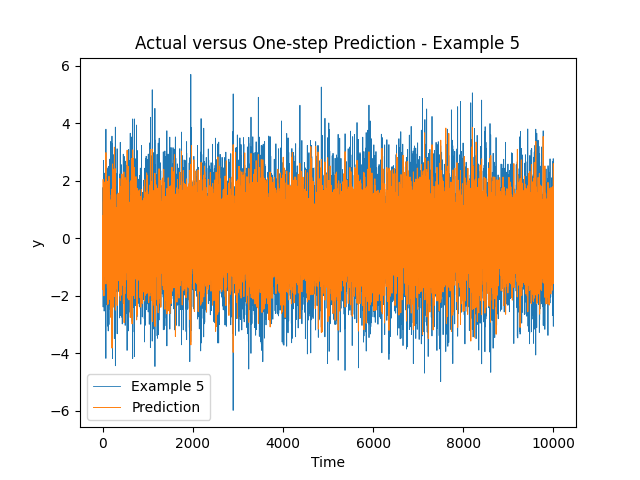
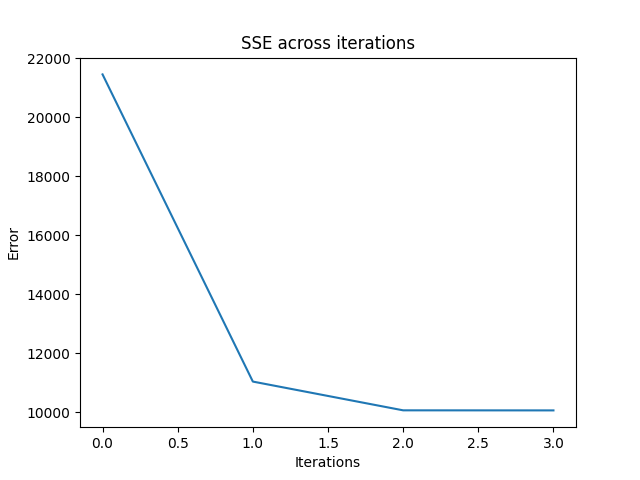
Example 3: ARMA (1,1): y(t) + 0.5y(t-1) = e(t) + 0.5e(t-1)



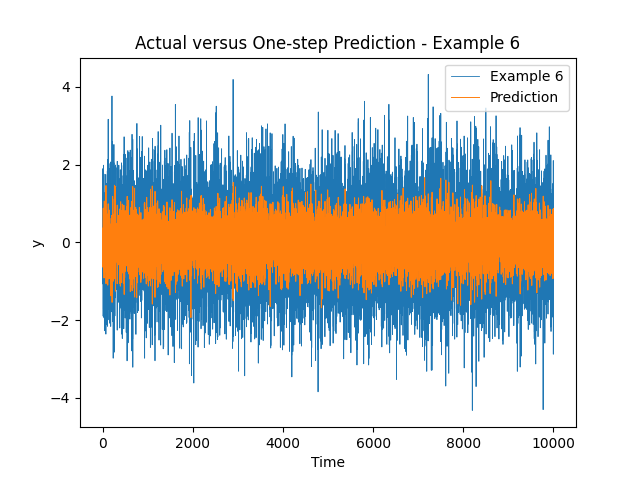
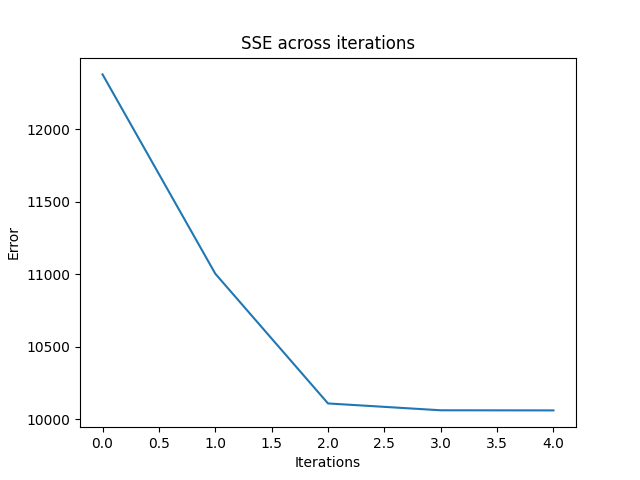
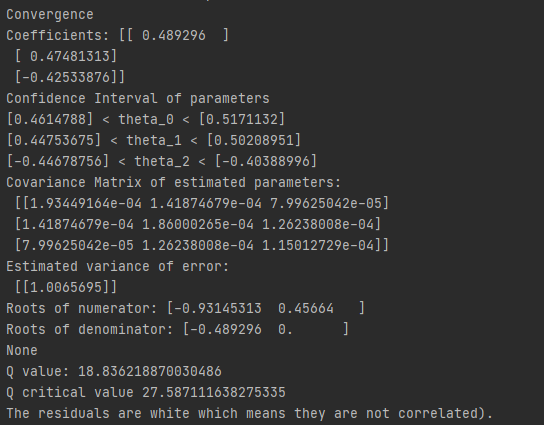
Example 4: ARMA (2,0): y(t) + 0.5y(t-1) + 0.2y(t-2) = e(t)



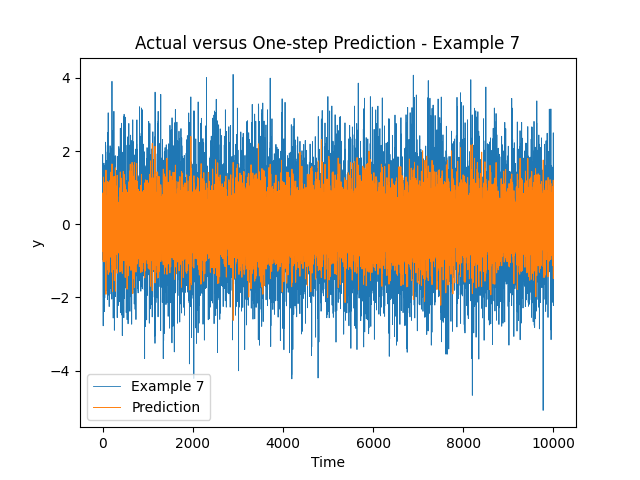
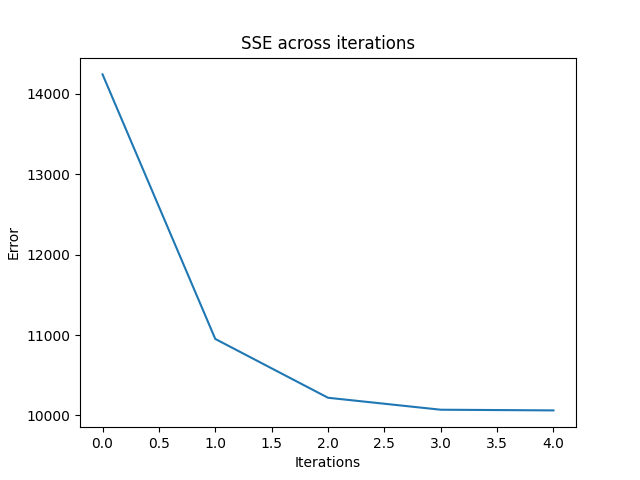
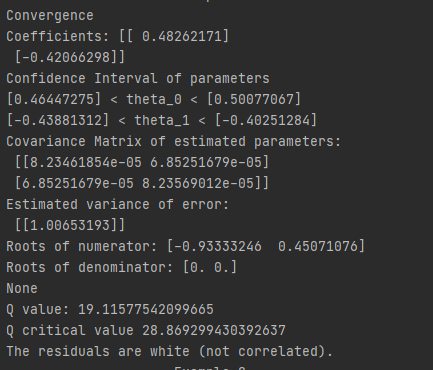
Example 5: ARMA (2,1): y(t) + 0.5y(t-1) + 0.2y(t-2) = e(t) - 0.5e(t-1)



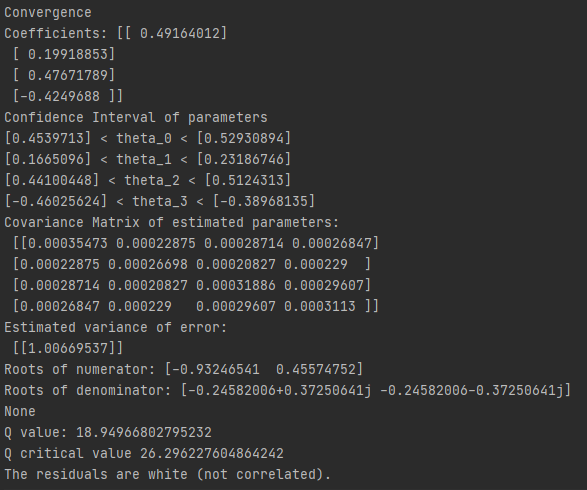
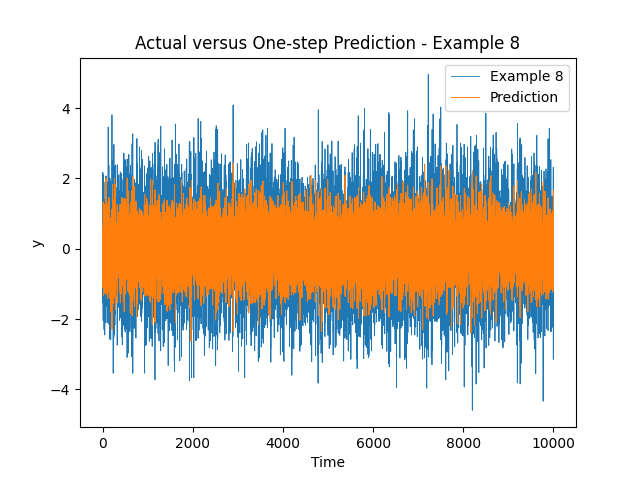
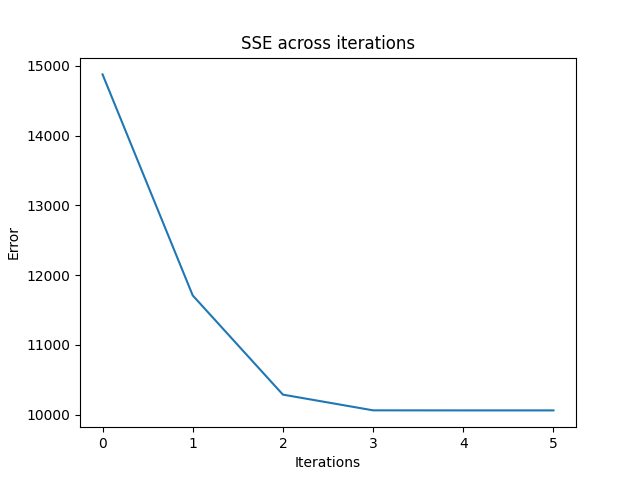
Example 6: ARMA (1,2): y(t) + 0.5y(t-1) = e(t) + 0.5e(t-1) - 0.4e(t-2)

****

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

****

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)



**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).")