func-test

2019 年 3 月 25 日

1　时间序列

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        from statsmodels.tsa.stattools import adfuller     # 平稳性检验
        import seaborn as sns
        import statsmodels.api as sm
        from statsmodels.stats.diagnostic import acorr_ljungbox
        from statsmodels.tsa.arima_model import ARMA
        import warnings
        from pywt import wavedec,waverec
        warnings.filterwarnings("ignore")
```

1.1　函数

```
In [2]: # 平稳性检验
        def adfTest(data):
            dftest = adfuller(data)
            if dftest[1] < 0.001:
                return True
            else:
                return False
```

```
In [3]: # ACF 和 PACF
        def acf_pacf_Test(data,n):
            acf,q,p = sm.tsa.acf(data,nlags=n,qstat=True)
            pacf = sm.tsa.acf(data,nlags=n)
            out = np.c_[range(1,n+1), acf[1:], pacf[1:], q, p]
            output=pd.DataFrame(out, columns=['lag', "AC", "PAC", "Q", "P-value"])
            return output
```

```
In [4]: # 返回各回归评估参数值 MSE,RMSE,MAE,R2
        def evaluationValue(data_true,data_predict):
            mse = np.sum((data_predict-data_true)**2) / len(data_true)
            rmse = np.sqrt(mse)
            mae = np.sum(np.absolute(data_predict-data_true)) / len(data_true)
            r2 = 1- mse / np.var(data_true)
            out = np.c_[1, mse, rmse, mae, r2]
            output=pd.DataFrame(out, columns=['index', "MSE", "RMSE", "MAE", "R2"])
            return output
```

```
In [5]: # 差分处理，默认最大为 5 阶
        def bestDiff(df, maxdiff = 6):
            temp = df.copy()
            first_values = []
            for i in range(0, maxdiff):
                if i == 0:
                    temp['diff'] = temp[temp.columns[0]]
                else:
                    first_values.append(pd.Series([temp['diff'][1]],index=[temp['diff'].index[0]])
                    temp['diff'] = temp['diff'].diff(1)
                    temp = temp.dropna() # 差分后，前几行的数据会变成 nan，所以删掉
                    # print(temp['diff'],'\n')
                if adfTest(temp['diff']):
                    bestdiff = i
                    return temp['diff'],first_values
                else:
                    continue
            return temp['diff'],first_values
```

```
In [6]: # 差分恢复
        def recoverDiff(df_diff,first_values):
            df_restored = df_diff
            for first in reversed(first_values):
                df_restored = first.append(df_restored).cumsum()
            return df_restored
```

```
In [7]: # HP 分解
        def hpFilter(data,l=1600):
            cycles, trend = sm.tsa.filters.hpfilter(data,l)
            return cycles,trend
```

```
In [8]: # DW 检验
```

```
def evaluationDW(resid):
    return sm.stats.durbin_watson(resid)
```

```
In [9]: # 小波分解
        def waveletFilter(data,level,func='db4'):
            coeffs = wavedec(data, func, level=level)
            return coeffs
```

```
In [10]: # 小波恢复
         def recoverWavelet(coeffs,func='db4'):
             data = waverec(coeffs, func)
             return data
```

```
In [11]: # 模型评估, 滚动预测
         def evaluationModle(data,order):
             train_size = int(len(data) * 0.66)
             train, test = data[0:train_size], data[train_size:]
             history = [x for x in train]
             predictions = list()
             for t in range(len(test)):
                 model = ARMA(history, order=order)
                 model_fit = model.fit(disp=0)
                 yhat = model_fit.forecast()[0]
                 predictions.append(yhat[0])
                 history.append(test[t])
             error = evaluationValue(test, predictions)
             # print(predictions)
             sns.lineplot(np.array(list(range(len(predictions)))),np.array(predictions),color='r')
             sns.lineplot(np.array(list(range(len(test)))),test,color='b')
             plt.show()
             return error,predictions
```

```
In [12]: # 模型 PQ 选择, 方法 1 (使用评估参数)
         def chooseModels1(data, maxlag=5,method='MSE'):
             best_score, best_cfg = float("inf"), None
             for p in np.arange(maxlag):
                 for q in np.arange(maxlag):
                     order = (p,q)
                     try:
                         error,predictions = evaluationModle(data, order)
                         print(order, error[method].values)
                         if error[method].values < best_score:
                             best_score, best_cfg = error[method].values, order
                     except:
                         print(order,'error')
                         continue
             return best_score,best_cfg
```

```
In [13]: # 模型 PQ 选择，方法 2（使用 aic,bic,hqic）
         def chooseModels2(data, maxlag=5,method='aic'):
             best_score, best_cfg = float("inf"), None
             for p in np.arange(maxlag):
                 for q in np.arange(maxlag):
                     order = (p,q)
                     model = ARMA(data, order=order)
                     try:
                         results_ARMA = model.fit(disp=0)
                         if method == 'aic':
                             score = results_ARMA.aic
                         elif method == 'bic':
                             score = results_ARMA.bic
                         elif method == 'hqic':
                             score = results_ARMA.hqic
                         print(order, score)
                         if score < best_score:
                             best_score, best_cfg = score, order
                     except:
                         print(order, 'error')
                         continue
             return best_score,best_cfg
```

```
In [14]: # 模型拟合
         def fitModel(data,order):
             model = ARMA(data,order=order)
             model_result = model.fit(disp=0)
             return model,model_result
```

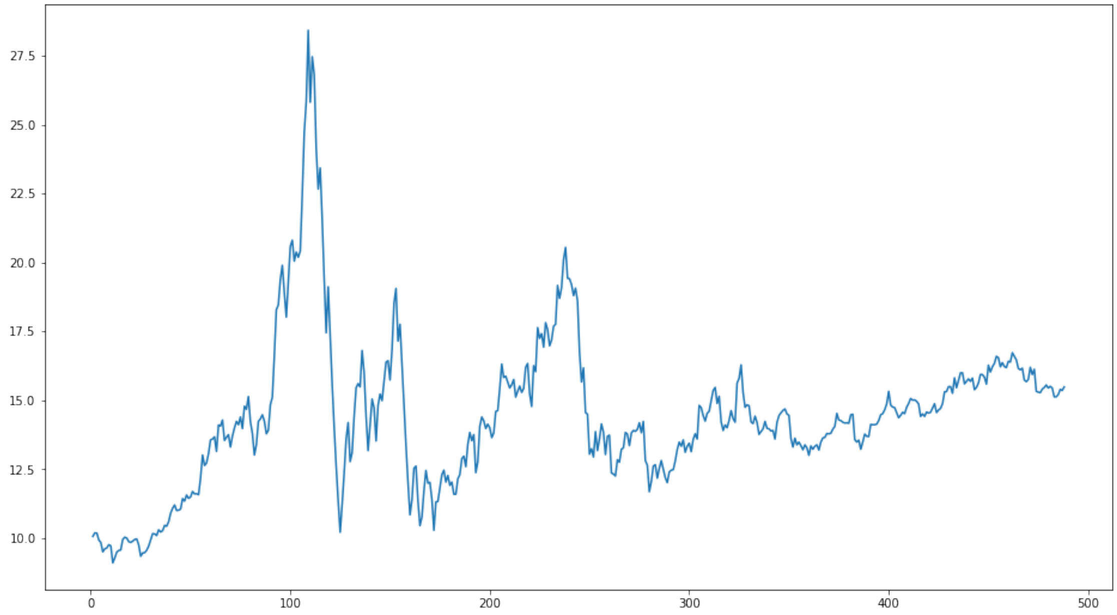
```
In [15]: # 模型检验
         def modelBLQ(model_result,n):
             output = acf_pacf_Test(results_ARMA.resid,n)['P-value']
             for i in range(len(output)):
                 if output[i] < 0.05:
                     return False
                 return True
```

```
In [16]: # 向内模型预测
         def forcastInModel(model_result):
             train_predict = model_result.predict()
             return train_predict
```

```
In [17]: # 向外模型预测
         def forcastOutModel(model_result,step):
             train_predict,b,train_predict_conf_int = model_result.forecast(step)
             return train_predict,train_predict_conf_int
```

```
In [18]: dicstock102419={"1":"10.07","2":"10.2","3":"10.19","4":"9.94","5":"9.85","6":"9.51","7":"
         x = np.array(list(dicstock102419.keys()),dtype='float64')
         y = np.array(list(dicstock102419.values()),dtype='float64')
         data = pd.DataFrame(y,x)
         plt.figure(figsize=(16,9))
         sns.lineplot(x,y)
```

```
Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x1c3d01c14a8>
```



1.2 差分序列

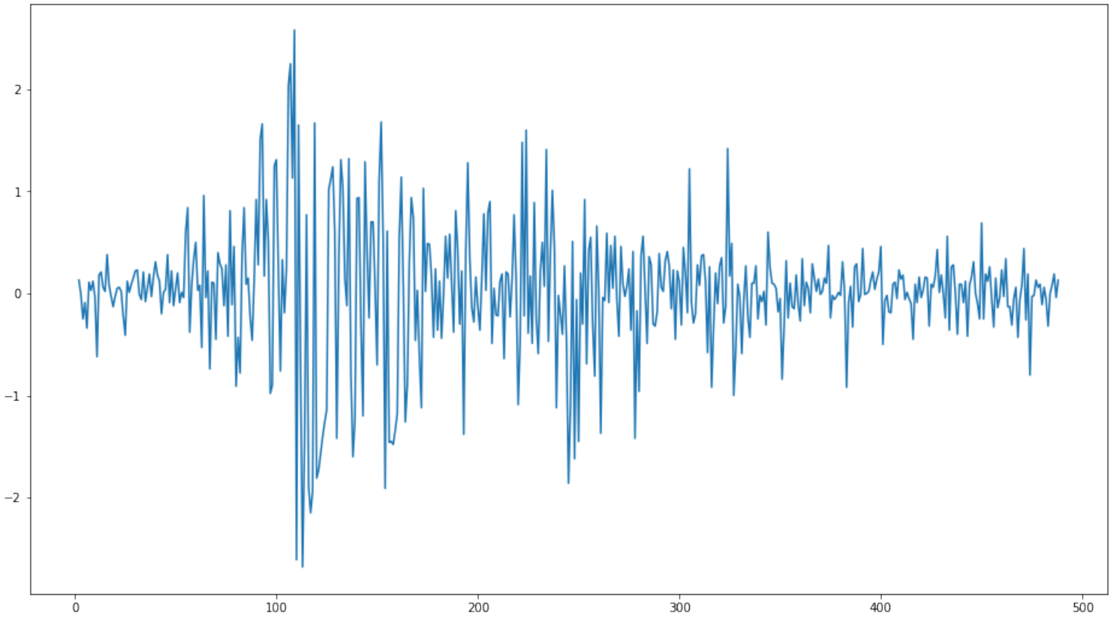
```
In [19]: adfTest(y)     # 不平稳
```

```
Out[19]: False
```

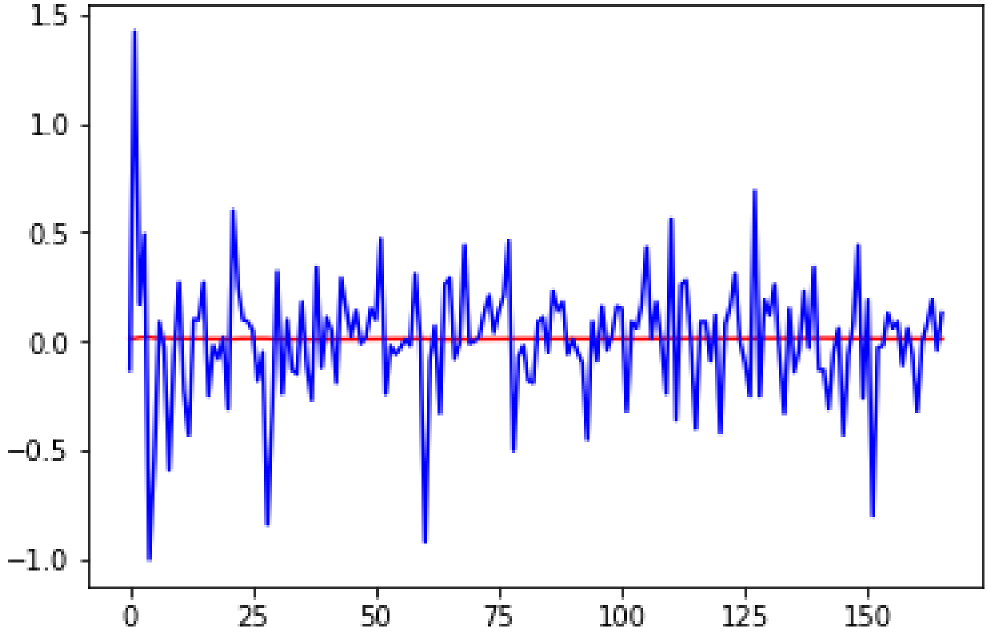
```
In [20]: datad,first_value = bestDiff(data)  # 进行差分, 差分阶数为 len(first_value)
```

```
In [21]: yd = datad.values
         plt.figure(figsize=(16,9))
         sns.lineplot(x[1:],yd)   # 差分后图像
```

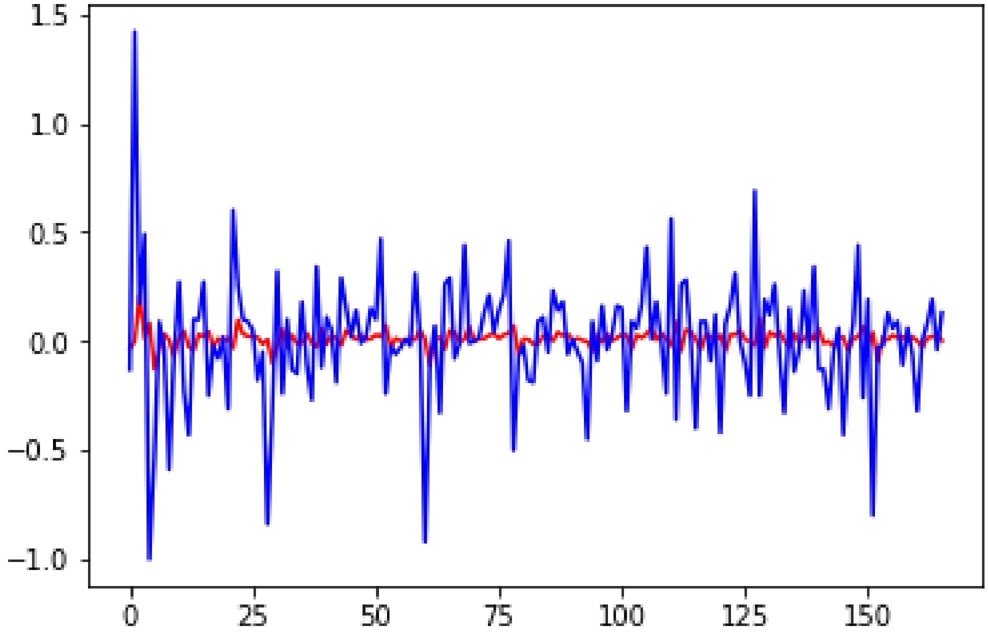
```
Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x1c3d77fb9b0>
```



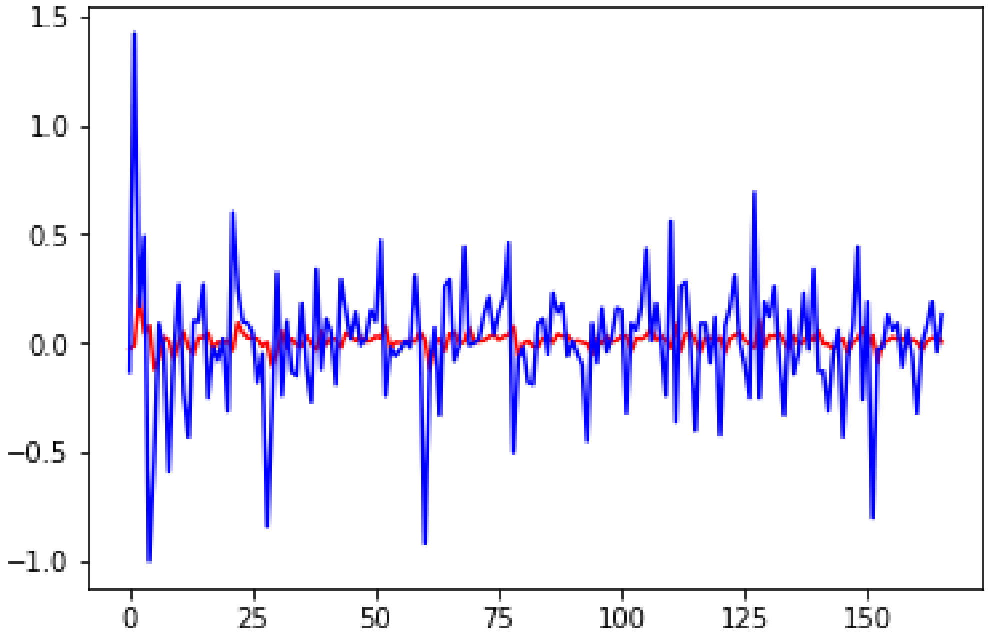
```
In [28]: best_score,best_cfg = chooseModels1(yd, maxlag=6,method='MSE')
```



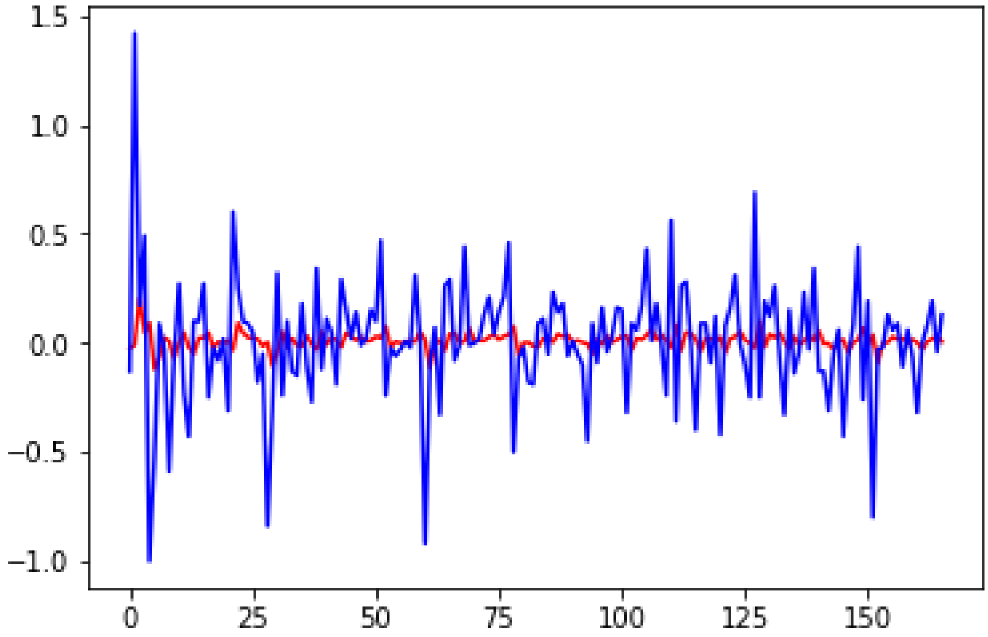
```
(0, 0) [0.08041686]
```



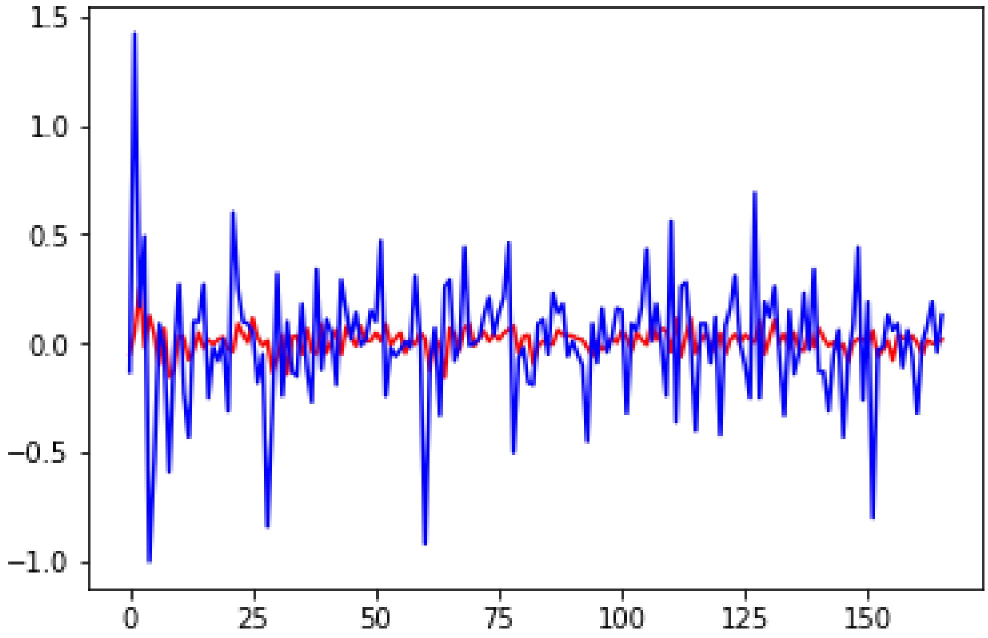
```
(0, 1) [0.08320272]
```



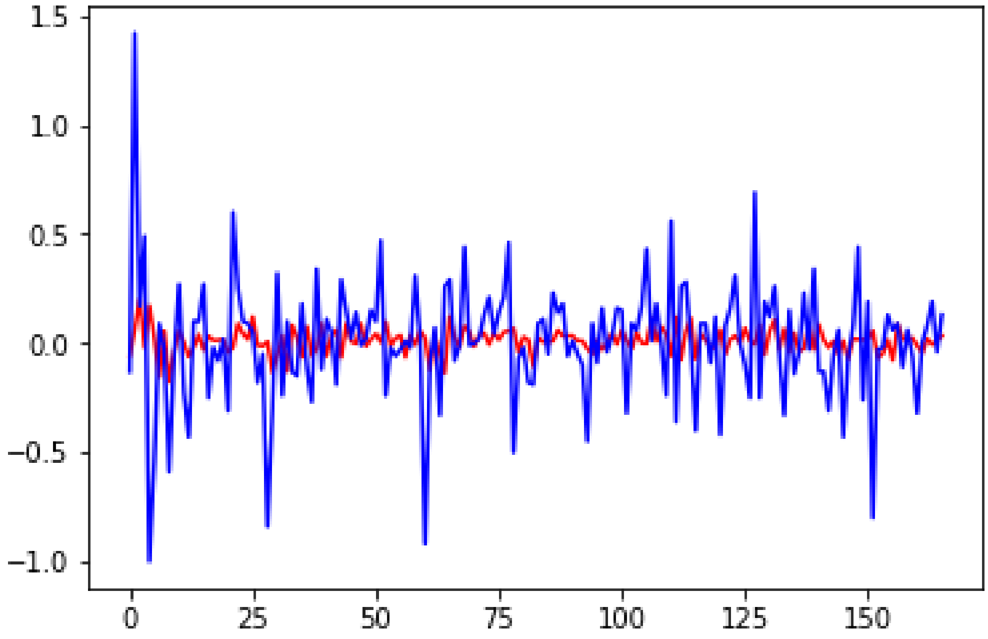
```
(0, 2) [0.08341979]
```



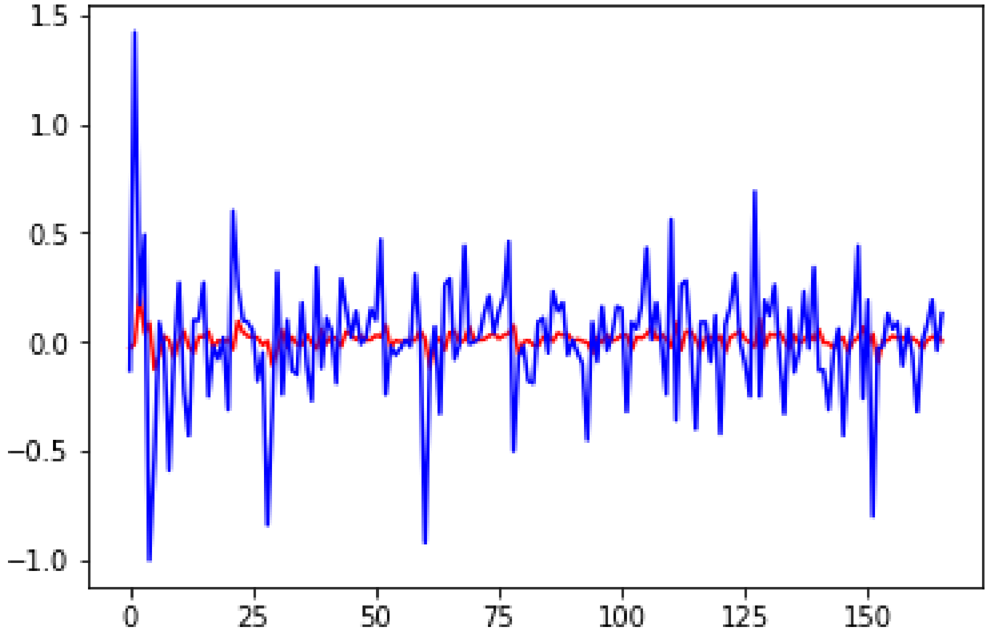
```
(0, 3) [0.08350787]
```



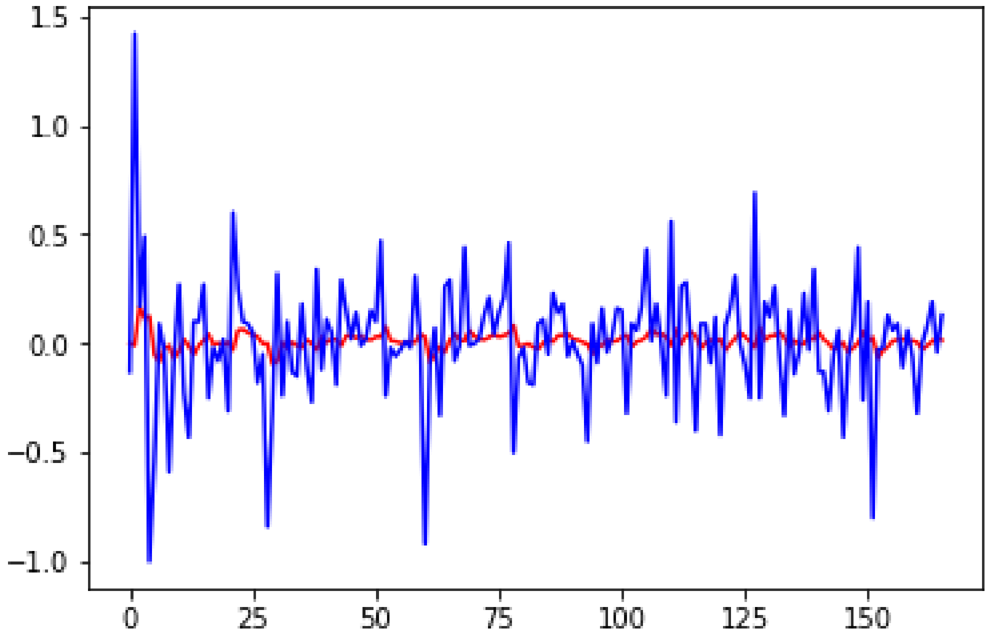
```
(0, 4) [0.08483243]
```



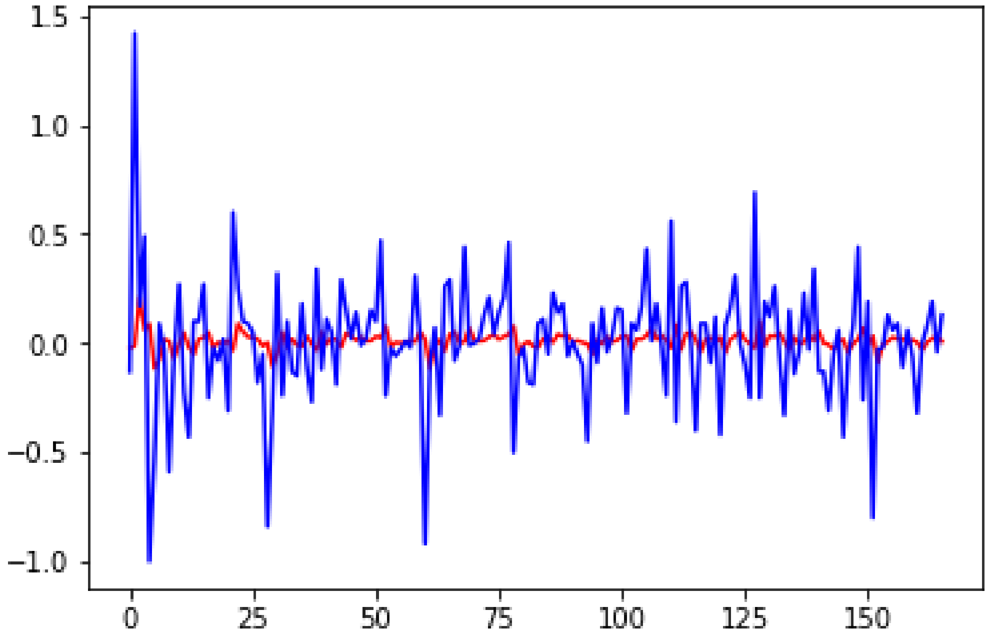
```
(0, 5) [0.08450419]
```



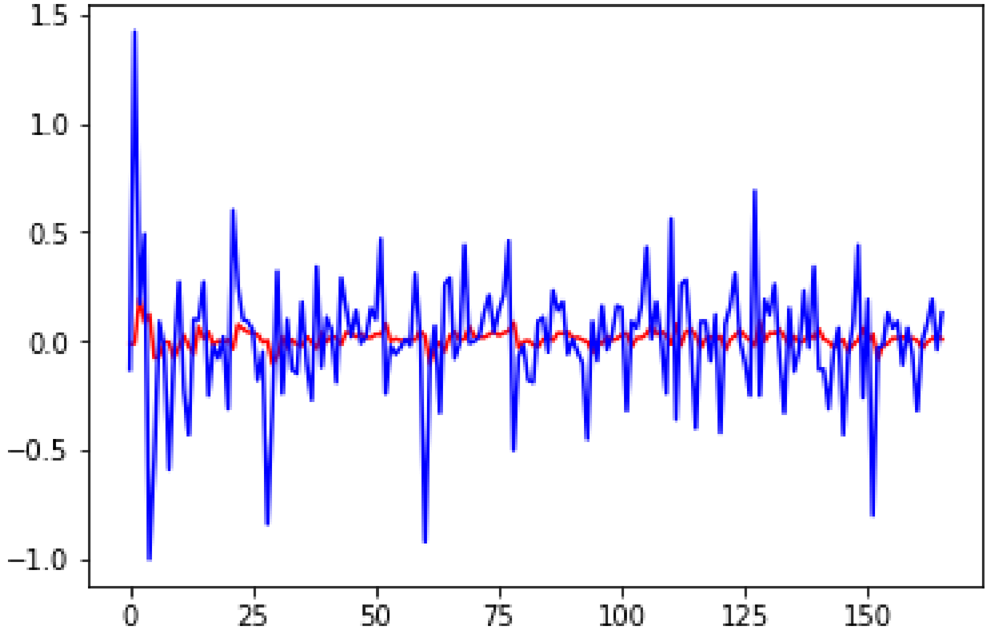
```
(1, 0) [0.08344131]
```



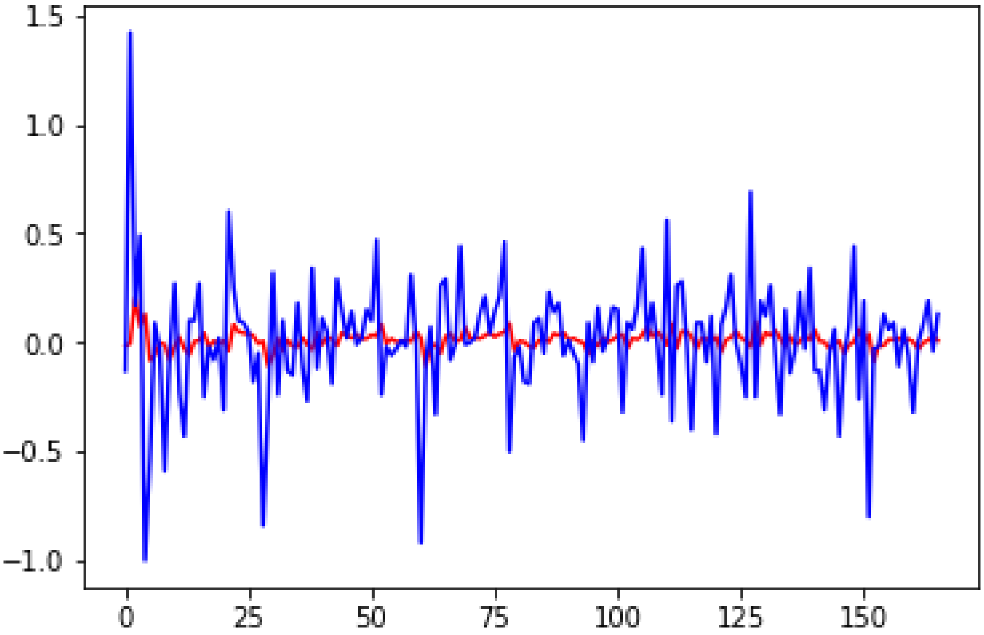
```
(1, 1) [0.0836965]
(1, 2) error
(1, 3) error
(1, 4) error
(1, 5) error
```



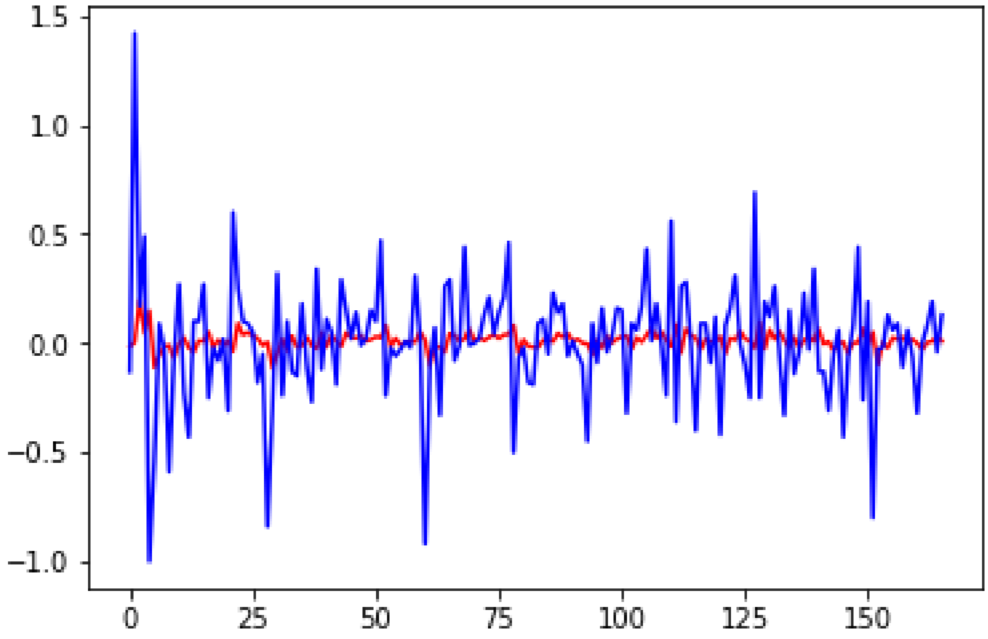
```
(2, 0) [0.08352119]
```



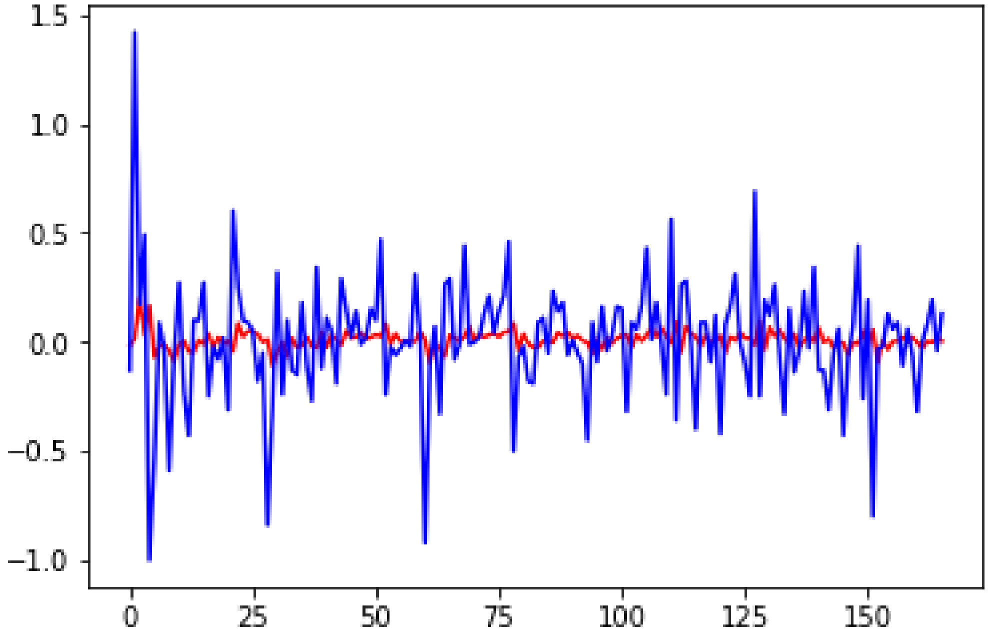
```
(2, 1) [0.08383703]
```



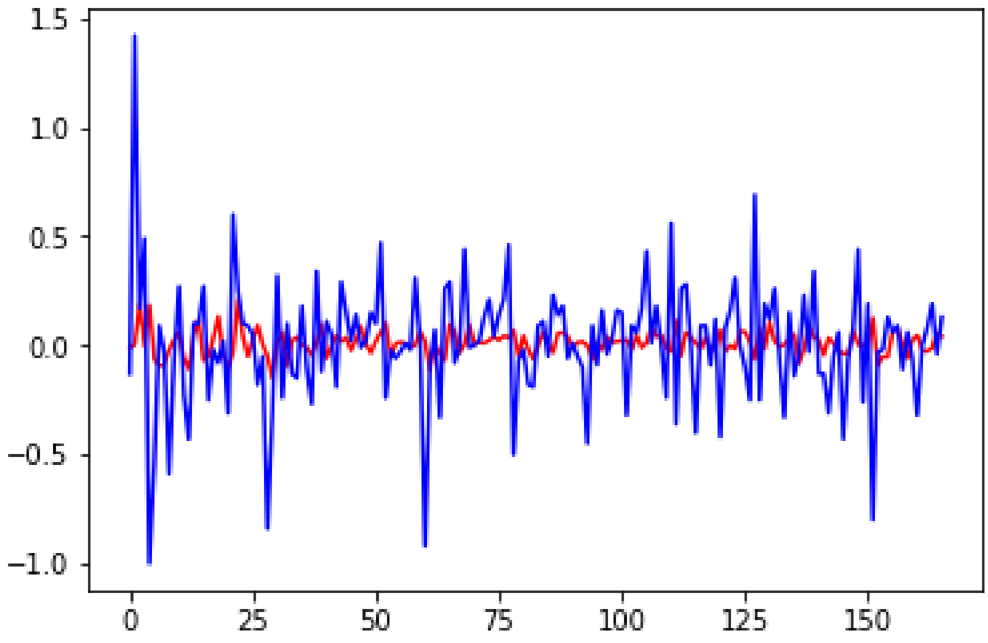
```
(2, 2) [0.08398973]
(2, 3) error
(2, 4) error
(2, 5) error
```



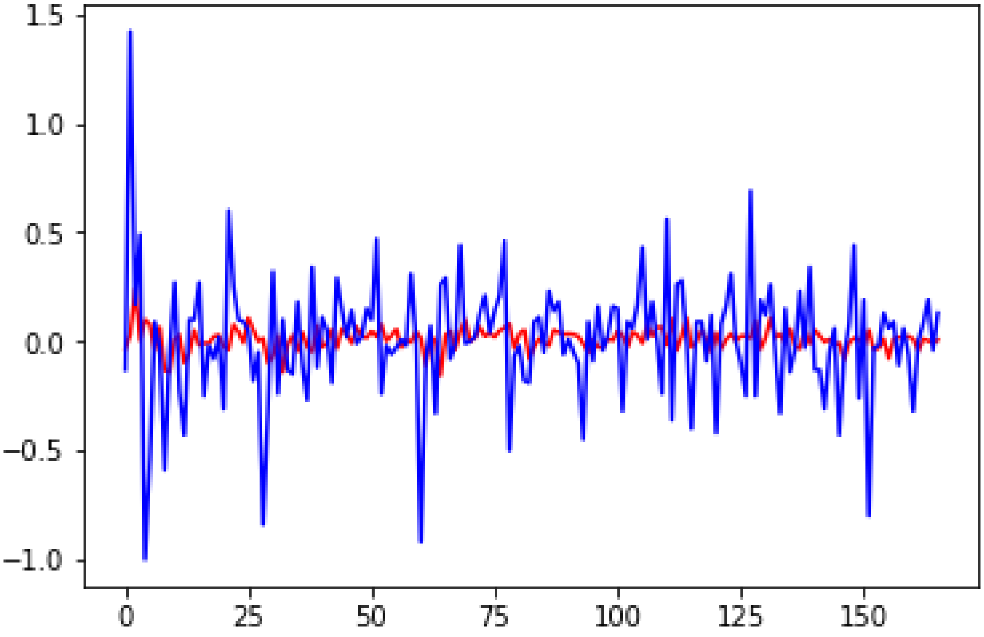
```
(3, 0) [0.08394737]
```



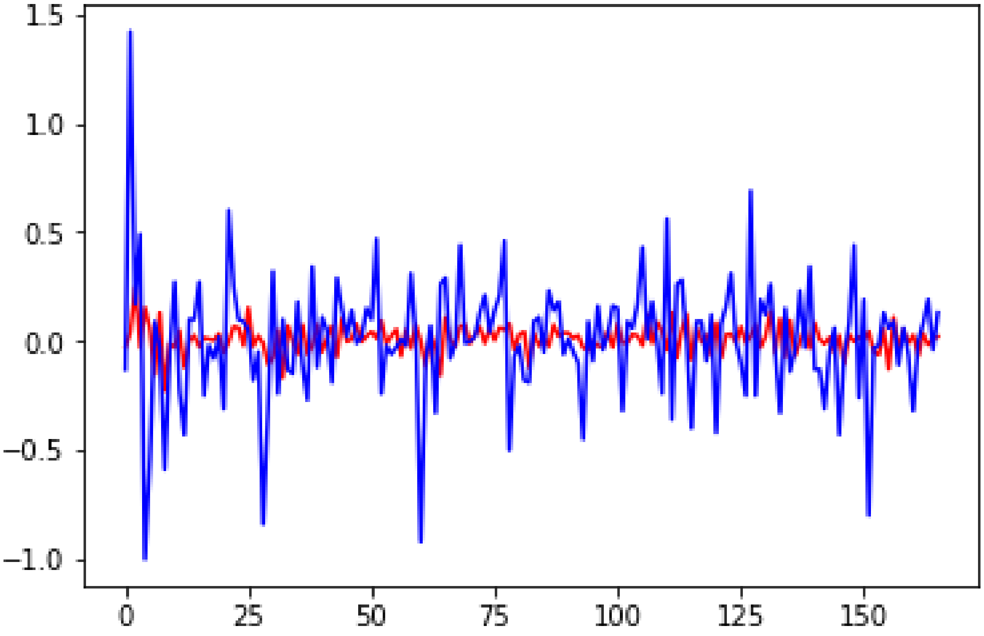
```
(3, 1) [0.08423015]
```



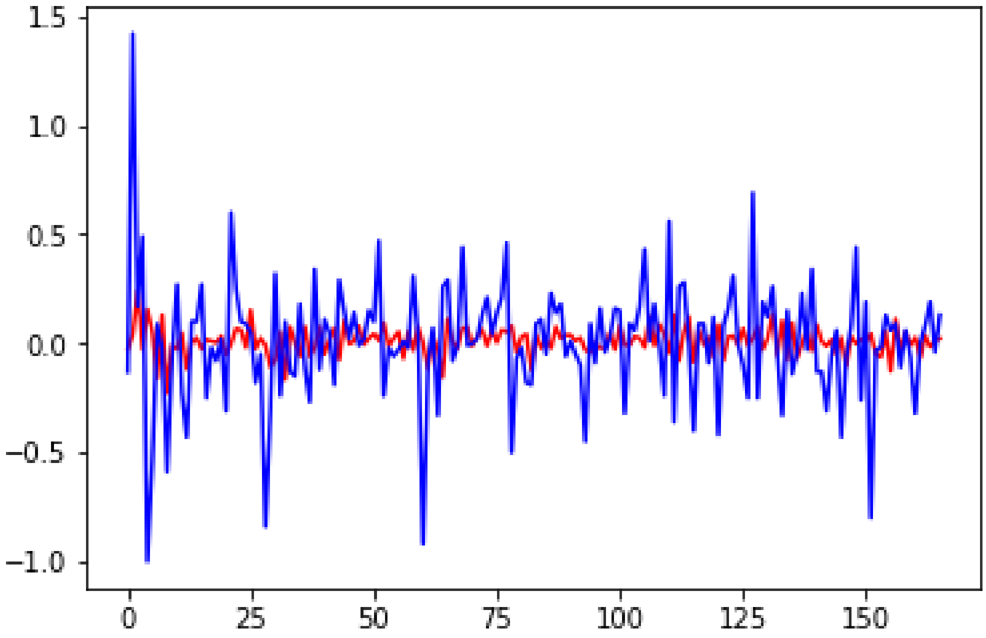
```
(3, 2) [0.08499342]
(3, 3) error
(3, 4) error
(3, 5) error
```



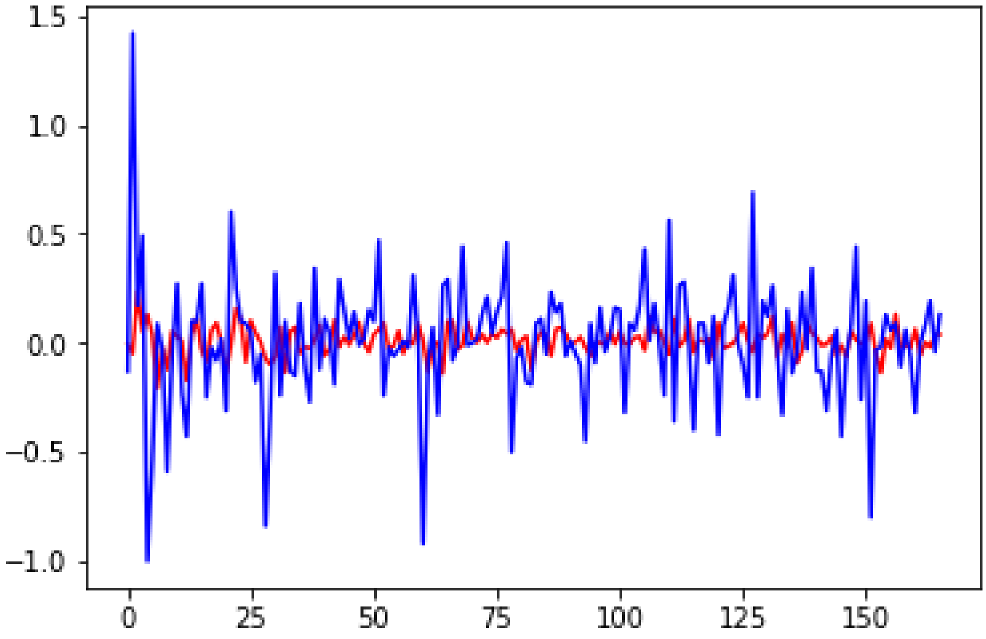
```
(4, 0) [0.08456955]
```



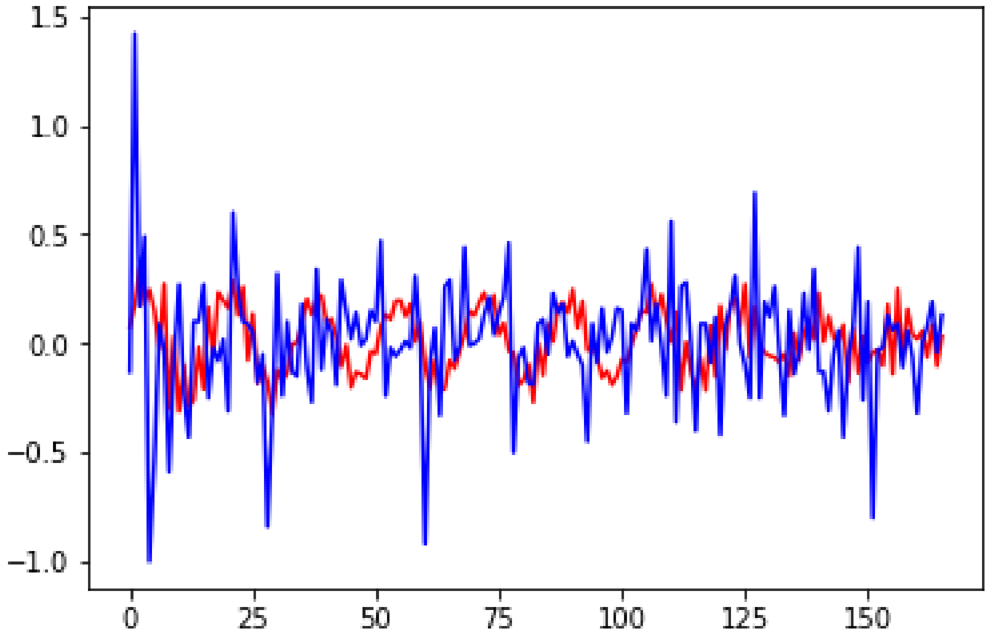
```
(4, 1) [0.08566541]
```



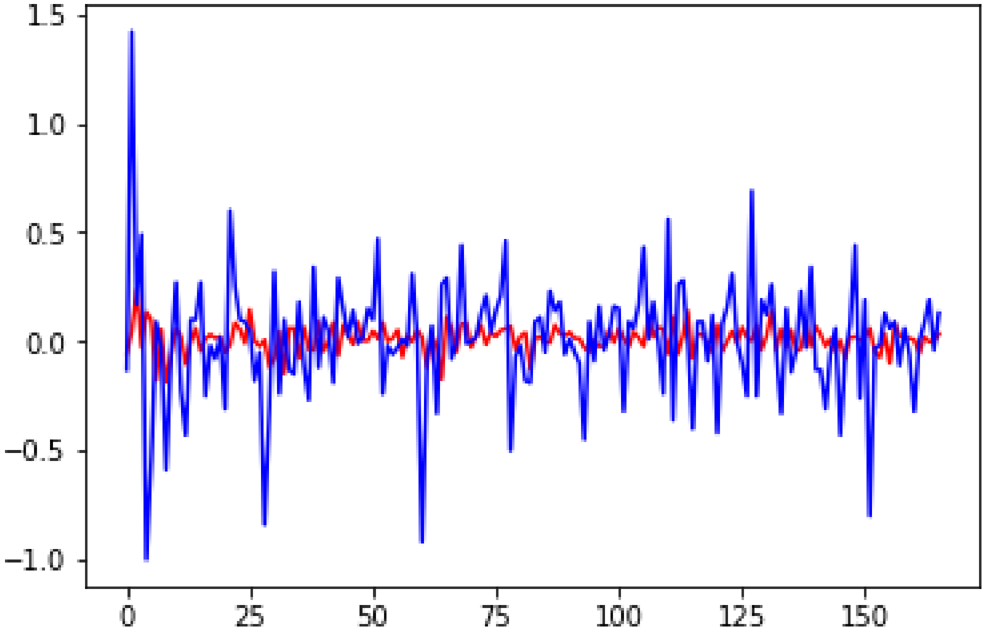
```
(4, 2) [0.085698]
```



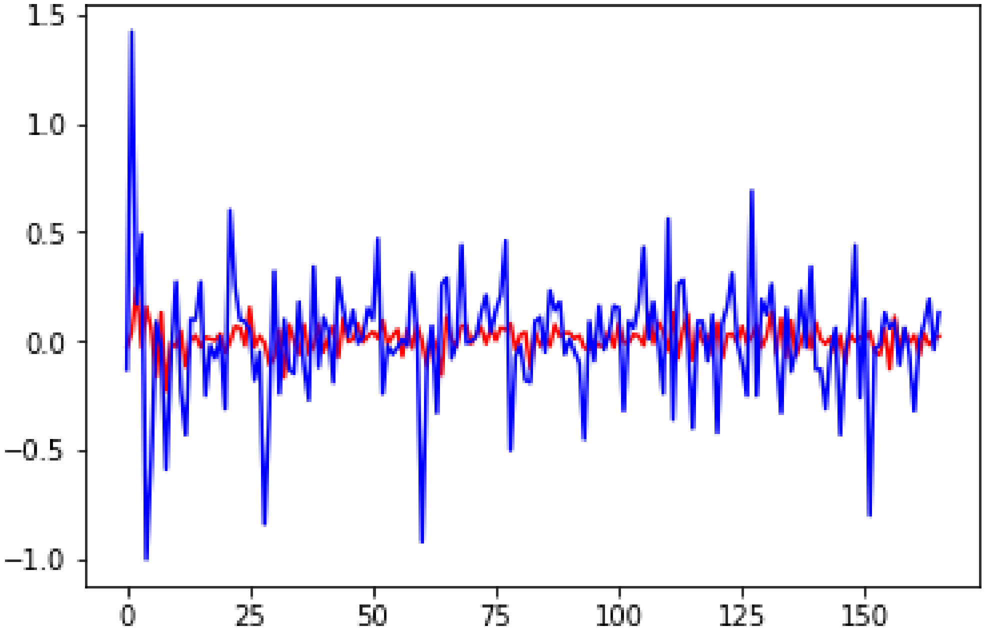
```
(4, 3) [0.08567986]
```



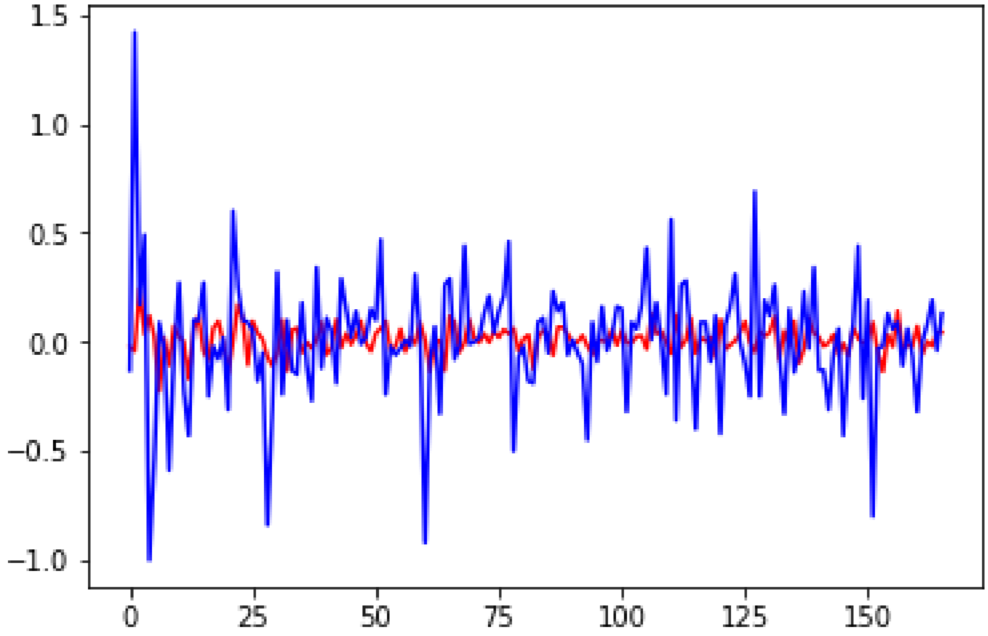
```
(4, 4) [0.09155374]
(4, 5) error
```



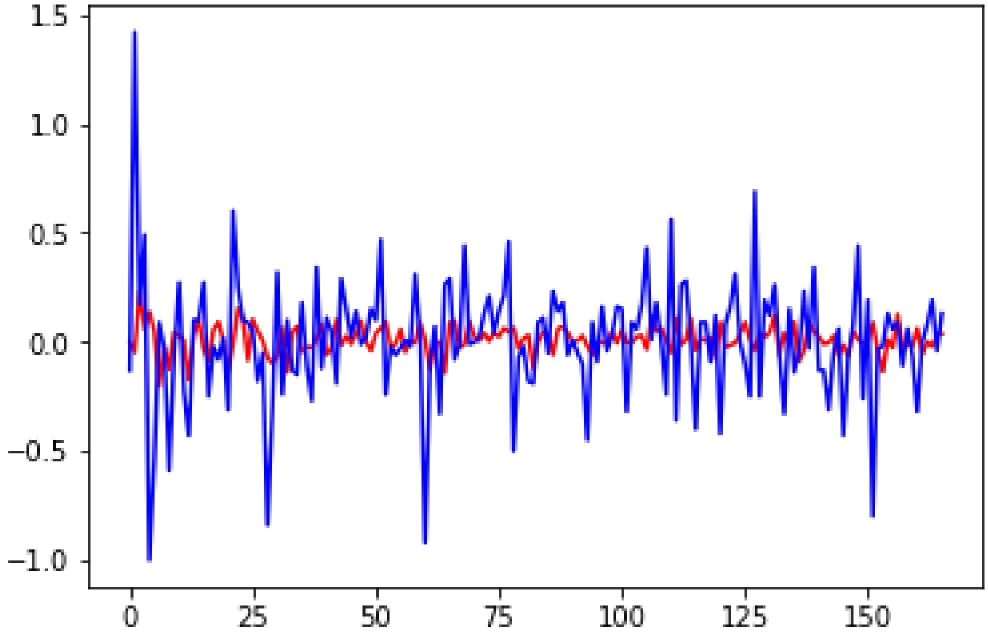
```
(5, 0) [0.08545094]
```



```
(5, 1) [0.08570125]
(5, 2) error
```



```
(5, 3) [0.08585186]
```



```
(5, 4) [0.08572857]
(5, 5) error
```

```
In [29]: best_score
```

```
Out[29]: array([0.08041686])
```

```
In [30]: best_cfg
```

```
Out[30]: (0, 0)
```

```
In [22]: best_score,best_cfg = chooseModels2(yd,5,method='bic')
```

```
(0, 0) 959.1172353315972
(0, 1) 957.4859273902542
(0, 2) 963.3907405000673
(0, 3) 969.5482164608977
(0, 4) 966.5943516648819
(1, 0) 957.1715060389663
(1, 1) 962.5312668374183
(1, 2) error
(1, 3) error
(1, 4) error
(2, 0) 963.2562614753615
(2, 1) 968.2489674079883
(2, 2) 974.2159309226545
(2, 3) error
(2, 4) error
(3, 0) 968.8001429399731
(3, 1) 973.2781191573879
(3, 2) 972.4588922607925
(3, 3) error
(3, 4) error
(4, 0) 967.2983227765253
(4, 1) 967.8405427806711
(4, 2) 974.016051893648
(4, 3) 977.9998034021537
(4, 4) 969.5956516007942
```

```
In [23]: best_score
```

```
Out[23]: 957.1715060389663
```

```
In [24]: best_cfg
```

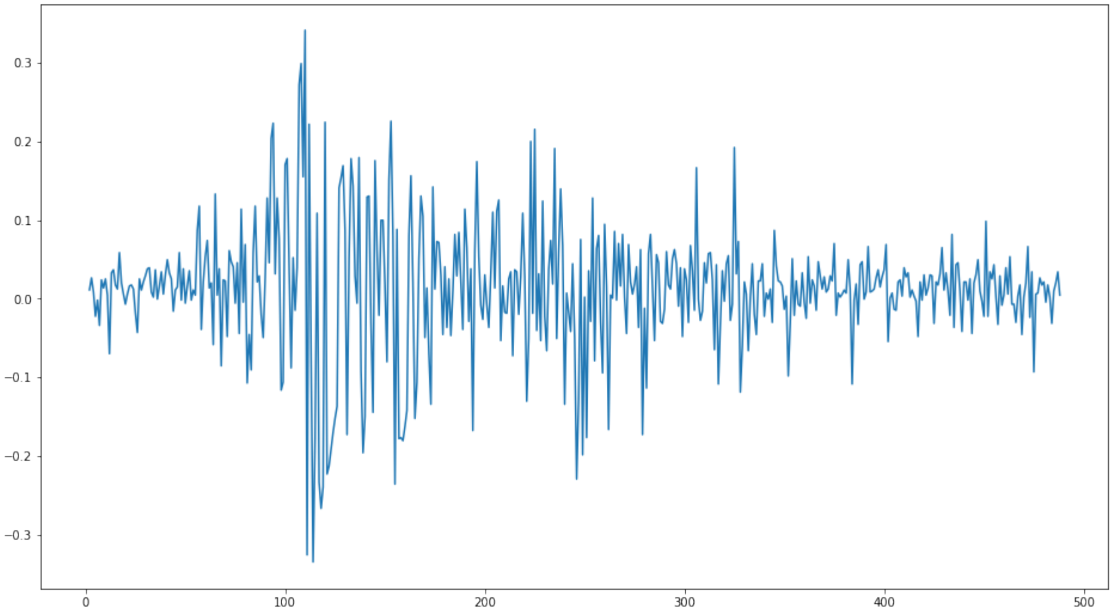
```
Out[24]: (1, 0)
```

```
In [25]: model, model_result = fitModel(yd,best_cfg)
```

```
In [26]: yp = forcastInModel(model_result)
```

```
In [27]: plt.figure(figsize=(16,9))
         sns.lineplot(x[1:],yp)     # 预测后图像
```

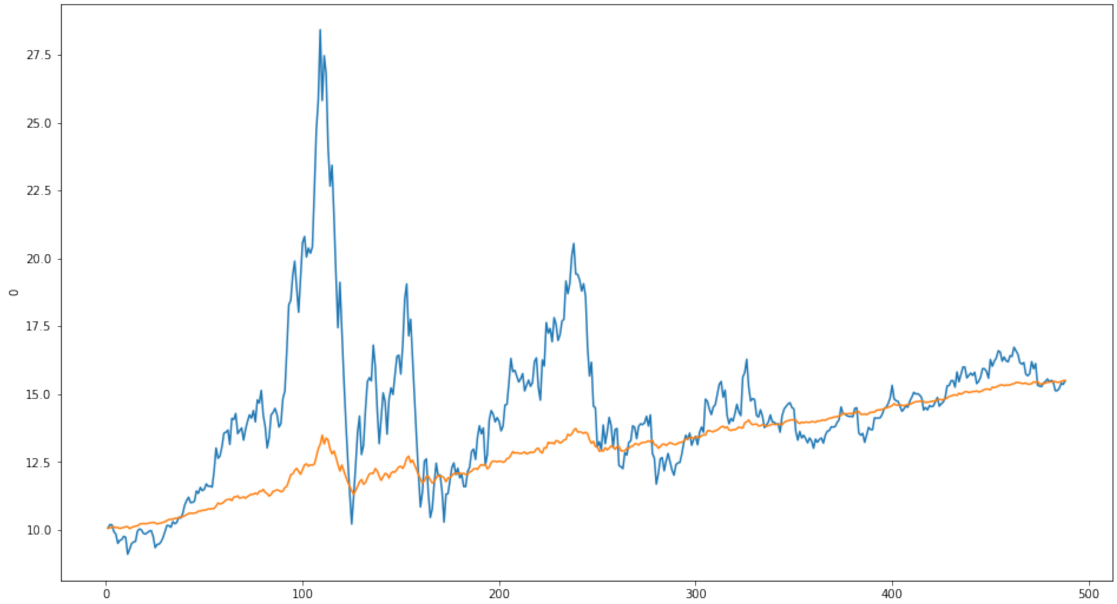
```
Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x1c3d7b7a0f0>
```



```
In [28]: df = pd.DataFrame(yp,index=list(range(len(first_value)+1,len(yp)+len(first_value)+1)))
         ypr = recoverDiff(df, first_value)
```

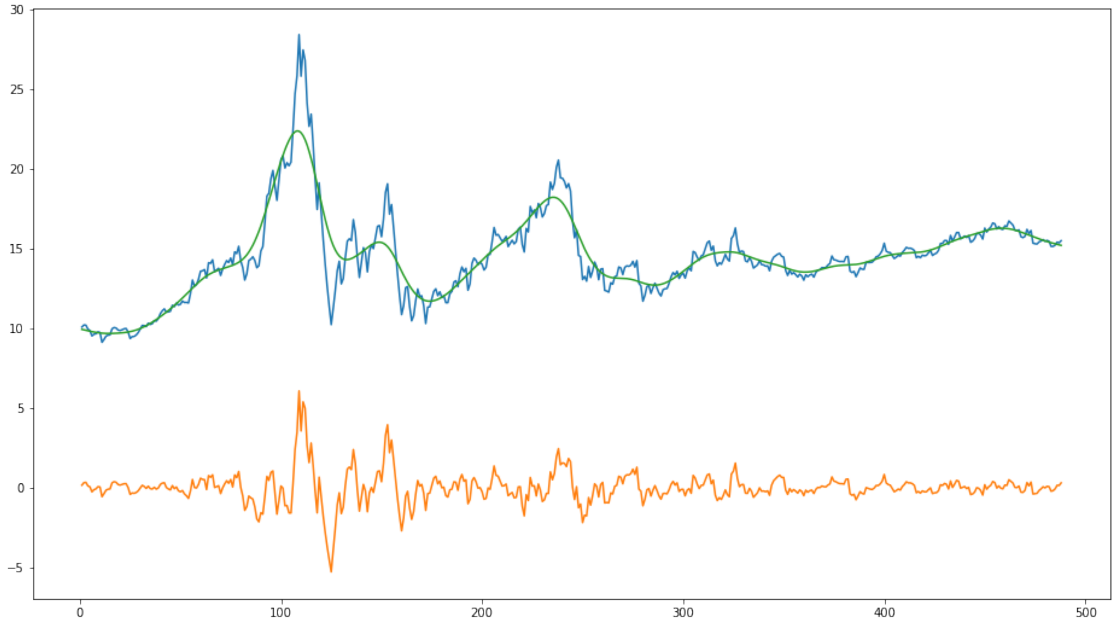
```
In [29]: plt.figure(figsize=(16,9))
         sns.lineplot(x,y)
         sns.lineplot(x,ypr[0])
```

```
Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0x1c3d7c00a90>
```

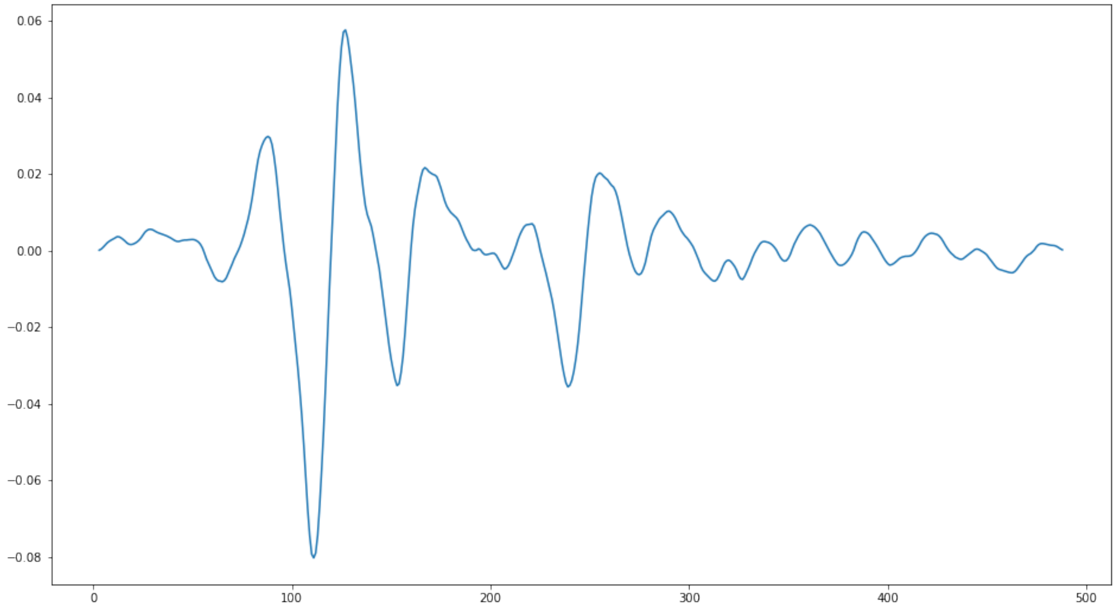


1.3 HP 滤波

```
In [30]: cycles,trend = hpFilter(y)     # 进行 HP 滤波分解
         plt.figure(figsize=(16,9))
         sns.lineplot(x,y)
         sns.lineplot(x,cycles)
         sns.lineplot(x,trend)
         step = 100;
```



```
In [31]: adfTest(cycles)

Out[31]: True

In [32]: adfTest(trend)

Out[32]: False

In [33]: td,tfv = bestDiff(pd.DataFrame(trend))

In [34]: td = td.values
         plt.figure(figsize=(16,9))
         sns.lineplot(x[len(tfv):],td)

Out[34]: <matplotlib.axes._subplots.AxesSubplot at 0x1c3d7b9c7b8>
```



```
In [35]: best_score,best_cfg = chooseModels2(td,5,method='bic')
```

```
(0, 0) -2598.043180875577
(0, 1) -3254.759511484198
(0, 2) error
(0, 3) error
(0, 4) error
(1, 0) -4366.111468724495
(1, 1) error
(1, 2) error
(1, 3) error
(1, 4) error
(2, 0) error
(2, 1) -6297.9330007461285
(2, 2) -6367.917773629811
(2, 3) -6367.76080941585
(2, 4) -6384.951528525135
(3, 0) error
(3, 1) -6389.767322626952
(3, 2) -6383.630828513481
(3, 3) -6377.769047619613
(3, 4) -6381.0497443139075
(4, 0) error
```

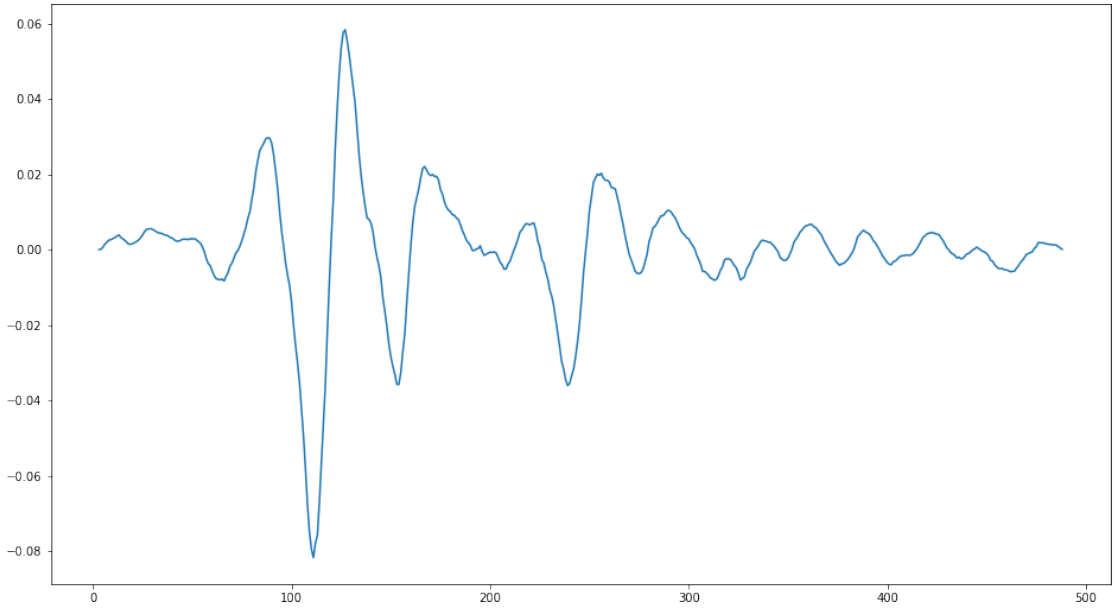
```
(4, 1) -6383.586663017474
(4, 2) -6383.510904787156
(4, 3) -6380.21594451777
(4, 4) -6378.893412652493
```

```
In [36]: best_score,best_cfg
```

```
Out[36]: (-6389.767322626952, (3, 1))
```

```
In [37]: model, model_result = fitModel(td,best_cfg)
         tp = forcastInModel(model_result)
         tpo,tpci = forcastOutModel(model_result,step)
         plt.figure(figsize=(16,9))
         sns.lineplot(x[len(tfv):],tp)
```

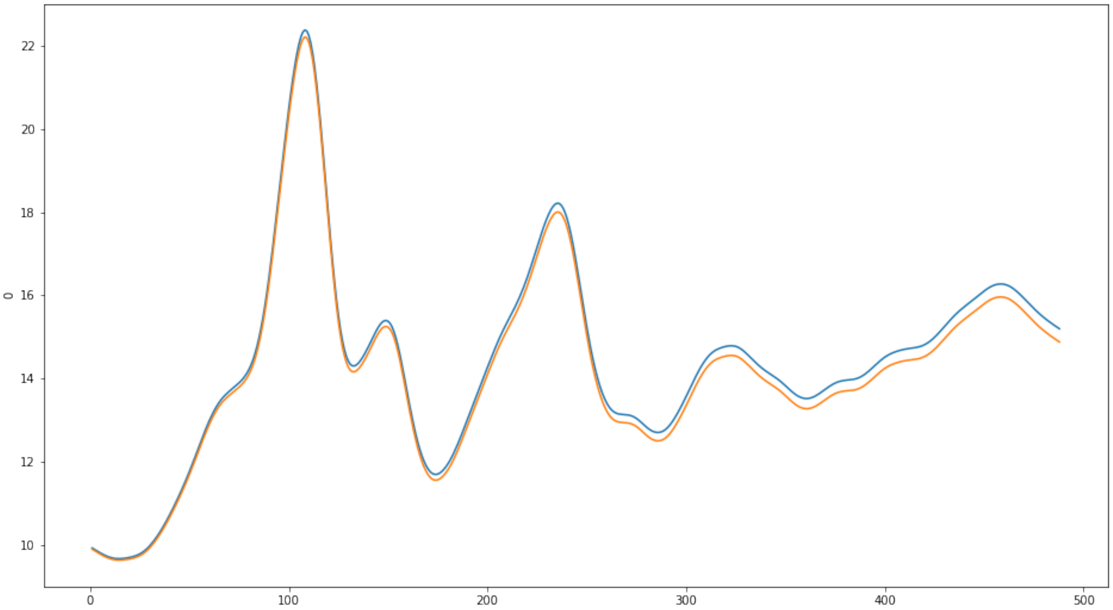
```
Out[37]: <matplotlib.axes._subplots.AxesSubplot at 0x1c3d7e997f0>
```



```
In [38]: df = pd.DataFrame(tp,index=list(range(len(tfv)+1,len(tp)+len(tfv)+1)))
         dfp = pd.DataFrame(np.append(tp,tpo),index=list(range(len(tfv)+1,len(tp)+len(tfv)+1+step)
         dfu = pd.DataFrame(np.append(tp,tpci[:,0]),index=list(range(len(tfv)+1,len(tp)+len(tfv)+1
         dfd = pd.DataFrame(np.append(tp,tpci[:,1]),index=list(range(len(tfv)+1,len(tp)+len(tfv)+1
         tpr = recoverDiff(df, tfv)
         tppr = recoverDiff(dfp, tfv)
```

```
        tpur = recoverDiff(dfu, tfv)
        tpdr = recoverDiff(dfd, tfv)
        plt.figure(figsize=(16,9))
        sns.lineplot(x,trend)
        sns.lineplot(x,tpr[0])
```

```
Out[38]: <matplotlib.axes._subplots.AxesSubplot at 0x1c3dc2619b0>
```



```
In [39]: best_score,best_cfg = chooseModels2(cycles,5,method='bic')
```

```
(0, 0) 1389.2153709162344
(0, 1) 1075.8705866617404
(0, 2) error
(0, 3) 918.4695979345088
(0, 4) error
(1, 0) 866.402359403205
(1, 1) 864.4100250597903
(1, 2) 869.7266713362823
(1, 3) 875.8923826627145
(1, 4) 869.1677978497584
(2, 0) 863.5411228112164
(2, 1) 812.4692202890211
(2, 2) 818.6543006755667
(2, 3) 821.2663930238037
```

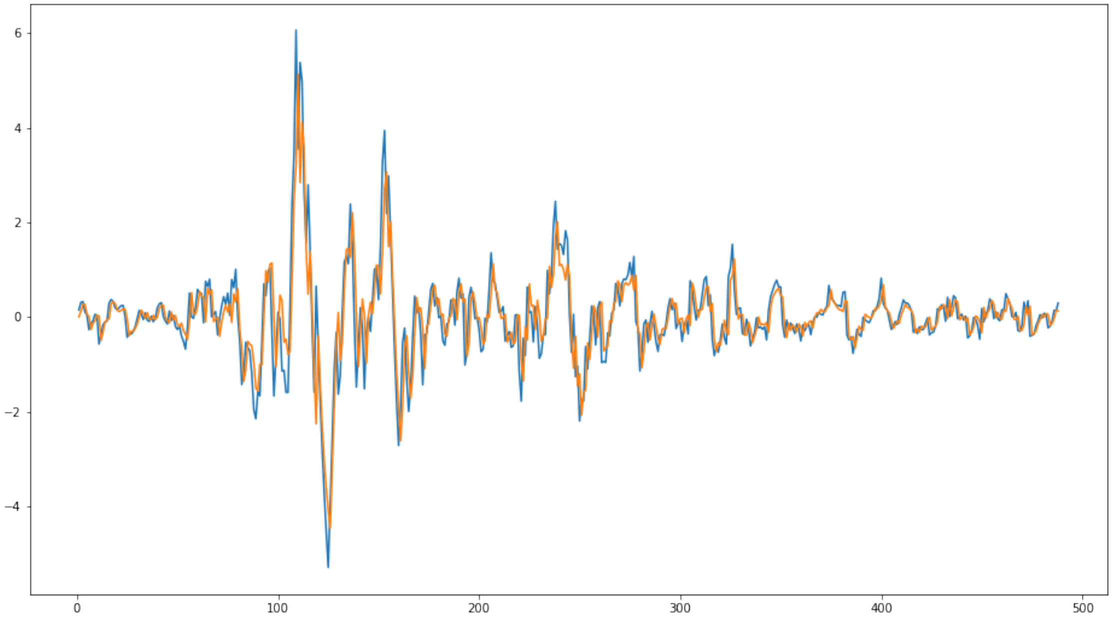
```
(2, 4) 823.6056709598388
(3, 0) 869.0027996275128
(3, 1) 818.6552737713404
(3, 2) 823.5266594353169
(3, 3) 828.8519616678823
(3, 4) 821.2537969513266
(4, 0) 873.2697039404278
(4, 1) 821.0011763720053
(4, 2) 829.2032881198581
(4, 3) 819.1717962916634
(4, 4) 822.7566863330409
```

```
In [40]: best_score,best_cfg
```

```
Out[40]: (812.4692202890211, (2, 1))
```

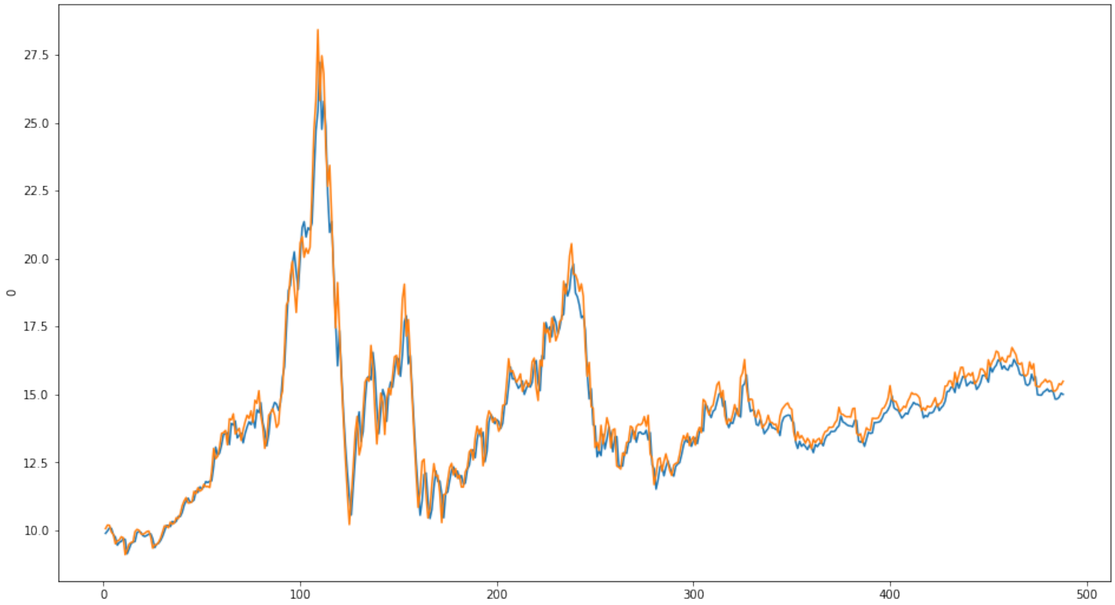
```
In [41]: model, model_result = fitModel(cycles,best_cfg)
         cp = forcastInModel(model_result)
         cpo,cpci = forcastOutModel(model_result,step)
         plt.figure(figsize=(16,9))
         sns.lineplot(x,cycles)
         sns.lineplot(x,cp)
```

```
Out[41]: <matplotlib.axes._subplots.AxesSubplot at 0x1c3dbb4a4a8>
```



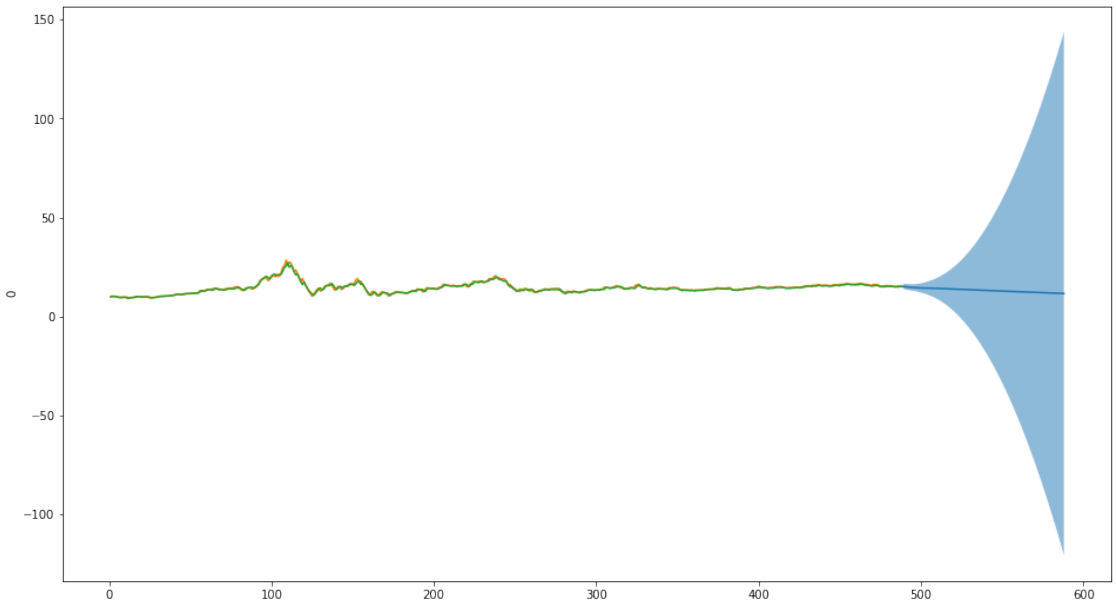
```
In [42]: pdata = cp + tpr[0]
         plt.figure(figsize=(16,9))
         sns.lineplot(x,pdata)
         sns.lineplot(x,y)
```

```
Out[42]: <matplotlib.axes._subplots.AxesSubplot at 0x1c3d4374048>
```



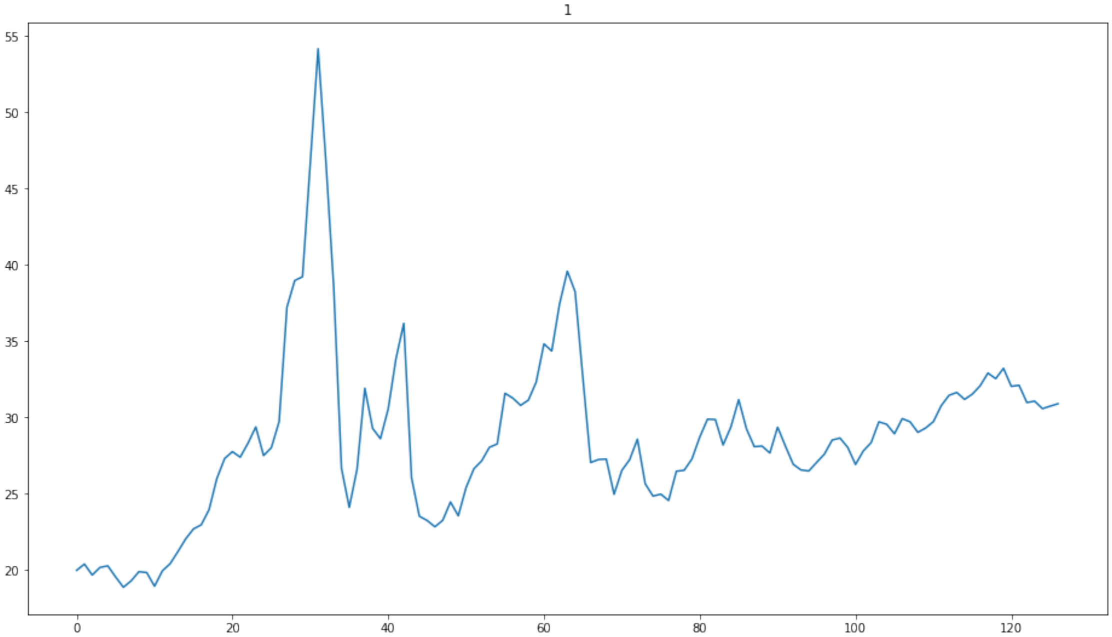
```
In [43]: pdatao = cpo + tppr[0][-step:].values
         plt.figure(figsize=(16,9))
         sns.lineplot(list(range(489,489+step)),pdatao)
         sns.lineplot(x,y)
         sns.lineplot(x,pdata)
         plt.fill_between(list(range(489,489+step)),cpci[:,0]+tpur[0][-step:].values,cpci[:,1]+tpd
```

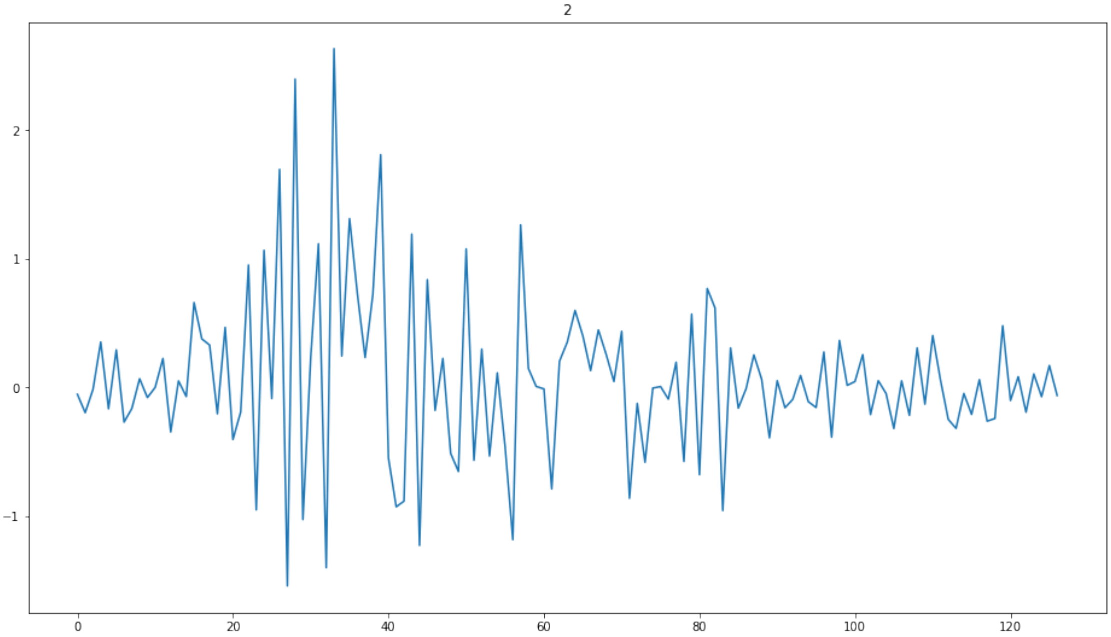
```
Out[43]: <matplotlib.collections.PolyCollection at 0x1c3dca05390>
```

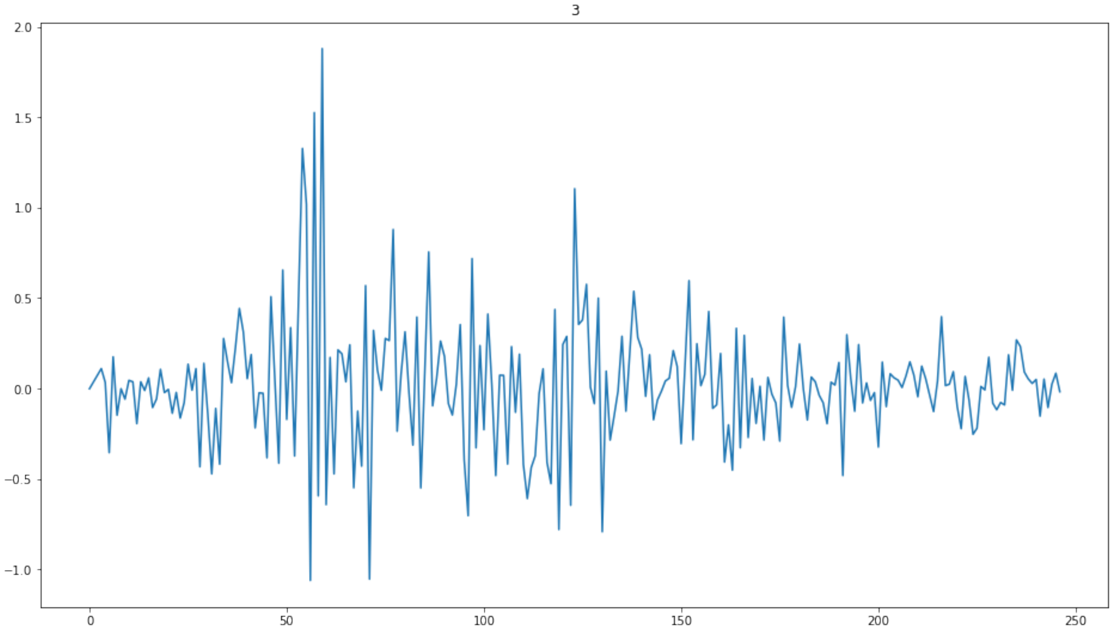


1.4 小波分析

```
In [44]: step = 100;
         coffes = waveletFilter(y,2)    # 进行 HP 滤波分解
         # plt.figure(figsize=(16,9))
         # sns.lineplot(x,y)
         coffes2 = waveletFilter(np.append(y,np.linspace(0,100,100)),2)
         steps = [len(coffes2[0])-len(coffes[0]),len(coffes2[1])-len(coffes[1]),len(coffes2[2])-le
         i = 1
         for coffe in coffes:
             # print(coffe)
             plt.figure(figsize=(16,9))
             plt.title(i)
             sns.lineplot(list(range(len(coffe))),coffe)
             i+=1
```


1
55
50
45
40
35
30
25
20
0
20
40
60
80
100
120


2
2
1
0
−1
0
20
40
60
80
100
120



```
In [45]: steps
```

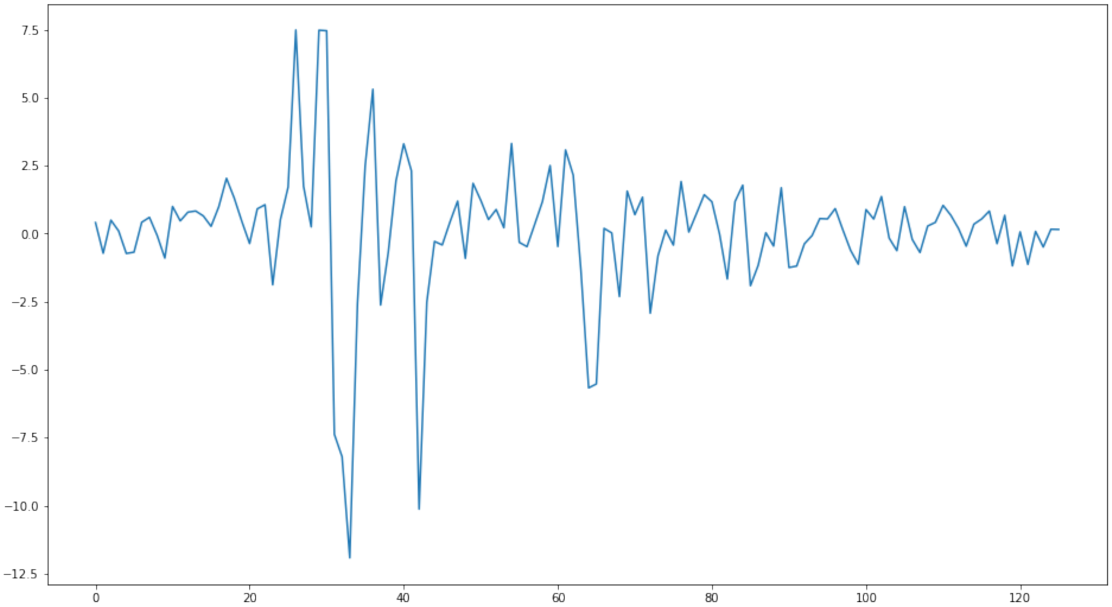
```
Out[45]: [25, 25, 50]
```

```
In [46]: for coffe in coffes:
             print(adfTest(coffe))
```

```
False
True
True
```

```
In [47]: ad,afv = bestDiff(pd.DataFrame(coffes[0]))
         ad = ad.values
         plt.figure(figsize=(16,9))
         sns.lineplot(list(range(len(ad))),ad)
```

```
Out[47]: <matplotlib.axes._subplots.AxesSubplot at 0x1c3de838780>
```



```
In [48]: best_score,best_cfg = chooseModels2(ad,5,method='bic')
```

```
(0, 0) 600.2690213370469
(0, 1) 585.4940144229992
(0, 2) 589.955112319535
(0, 3) 589.9618183905318
(0, 4) 590.2239513506528
(1, 0) 589.642559131766
(1, 1) 590.1908674016591
(1, 2) 586.4888651264454
(1, 3) 587.6532425232247
(1, 4) error
(2, 0) 586.0944469320939
(2, 1) 586.6458716253658
(2, 2) 592.4930834250305
(2, 3) 592.354181000786
(2, 4) error
(3, 0) 589.0723202526272
(3, 1) 593.6017993920834
(3, 2) 595.5905190833057
(3, 3) 595.1443197782121
(3, 4) error
(4, 0) 593.6773551342563
```

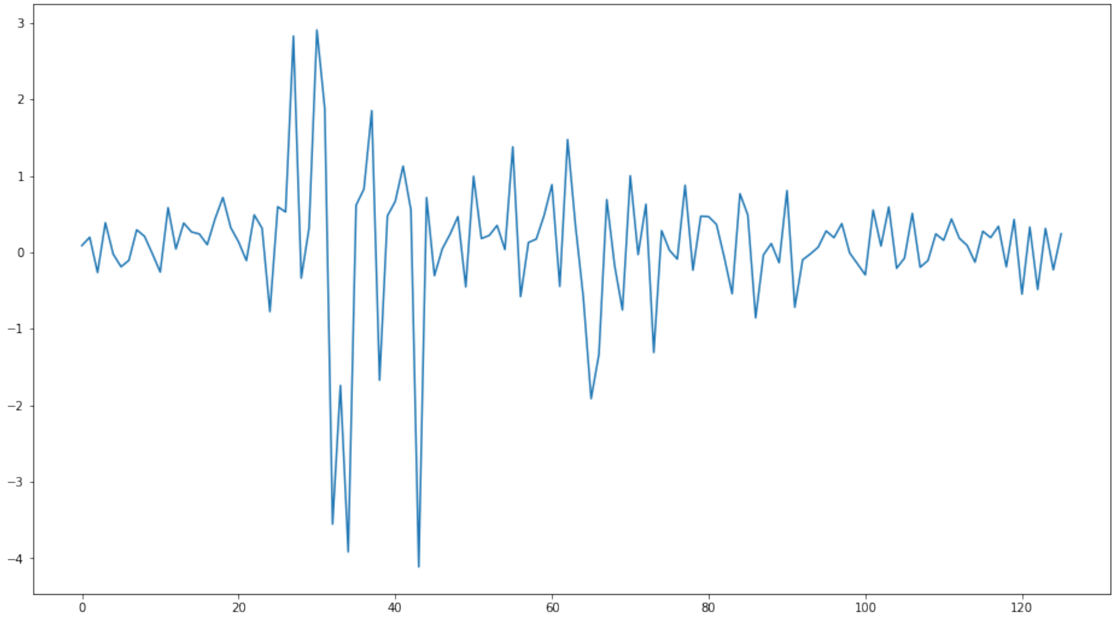
```
(4, 1) 598.4113021914732
(4, 2) 599.0435780078417
(4, 3) 595.0999120465101
(4, 4) error
```

```
In [49]: best_score,best_cfg
```

```
Out[49]: (585.4940144229992, (0, 1))
```

```
In [50]: model, model_result = fitModel(ad,best_cfg)
         ap = forcastInModel(model_result)
         apo,apci = forcastOutModel(model_result,steps[0])
         plt.figure(figsize=(16,9))
         sns.lineplot(list(range(len(ad))),ap)
```

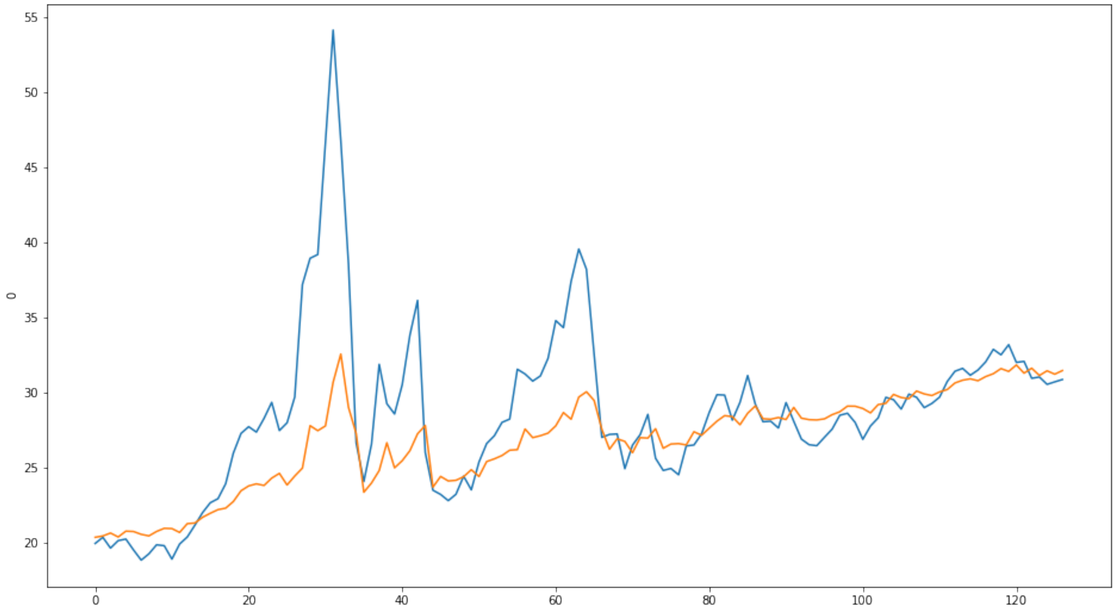
```
Out[50]: <matplotlib.axes._subplots.AxesSubplot at 0x1c3dd1baa20>
```



```
In [51]: df = pd.DataFrame(ap,index=list(range(len(afv)+1,len(ap)+len(afv)+1)))
         dfp = pd.DataFrame(np.append(ap,apo),index=list(range(len(afv)+1,len(ap)+len(afv)+1+steps
         dfu = pd.DataFrame(np.append(ap,apci[:,0]),index=list(range(len(afv)+1,len(ap)+len(afv)+1
         dfd = pd.DataFrame(np.append(ap,apci[:,1]),index=list(range(len(afv)+1,len(ap)+len(afv)+1
         apr = recoverDiff(df, afv)
         appr = recoverDiff(dfp, afv)
```

```
apur = recoverDiff(dfu, afv)
apdr = recoverDiff(dfd, afv)
plt.figure(figsize=(16,9))
sns.lineplot(list(range(len(coffes[0]))),coffes[0])
sns.lineplot(list(range(len(coffes[0]))),apr[0])
```

```
Out[51]: <matplotlib.axes._subplots.AxesSubplot at 0x1c3de841ef0>
```



```
In [52]: best_score,best_cfg = chooseModels2(coffes[1],5,method='bic')
```

```
(0, 0) 256.8877896759146
(0, 1) 246.60321345988416
(0, 2) 238.54681383591608
(0, 3) 242.4320592760104
(0, 4) 246.40345707547652
(1, 0) 238.75025101238737
(1, 1) 241.8895795959282
(1, 2) 242.24665212356945
(1, 3) 247.92504128502316
(1, 4) error
(2, 0) 240.23701142961139
(2, 1) 242.03960340450195
(2, 2) 246.73250065474633
(2, 3) 250.9756792940887
```

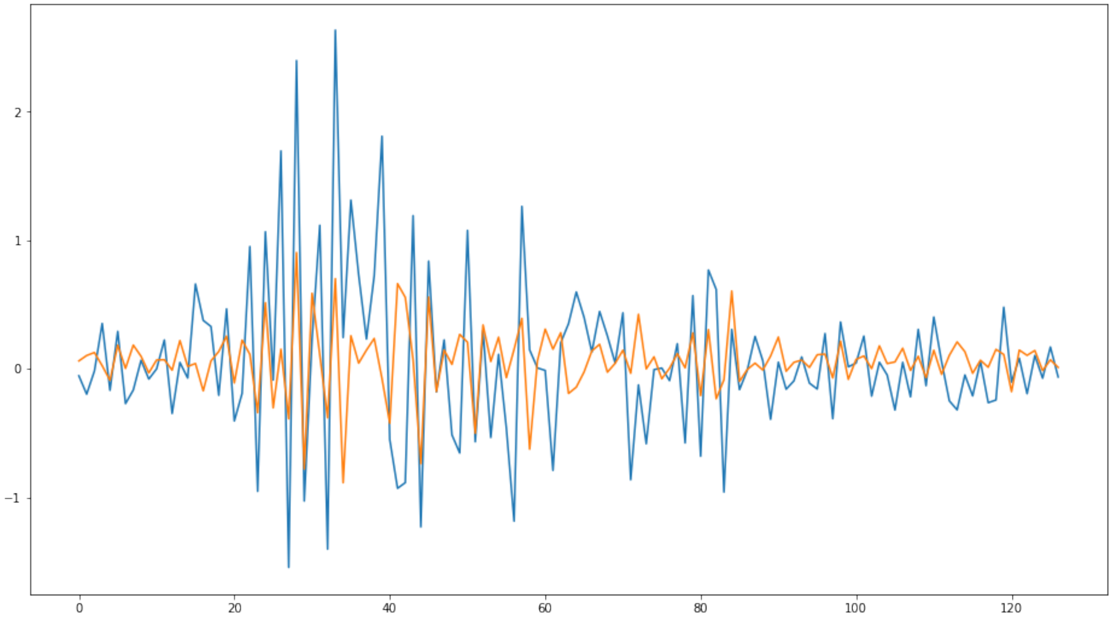
```
(2, 4) error
(3, 0) 241.1703275160015
(3, 1) 239.7856183092954
(3, 2) 243.19634824979374
(3, 3) 247.86181364202332
(3, 4) 255.96318675030494
(4, 0) 245.6051924636973
(4, 1) 243.52734764366147
(4, 2) 247.80261358413503
(4, 3) 252.6311748588194
(4, 4) 256.4190894695959
```

```
In [53]: best_score,best_cfg
```

```
Out[53]: (238.54681383591608, (0, 2))
```

```
In [54]: model, model_result = fitModel(coffes[1],best_cfg)
         dp1 = forcastInModel(model_result)
         dpo1,dpci1 = forcastOutModel(model_result,steps[1])
         plt.figure(figsize=(16,9))
         sns.lineplot(list(range(len(coffes[1]))),coffes[1])
         sns.lineplot(list(range(len(dp1))),dp1)
```

```
Out[54]: <matplotlib.axes._subplots.AxesSubplot at 0x1c3dc95acc0>
```



```
In [55]: best_score,best_cfg = chooseModels2(coffes[2],5,method='bic')
```

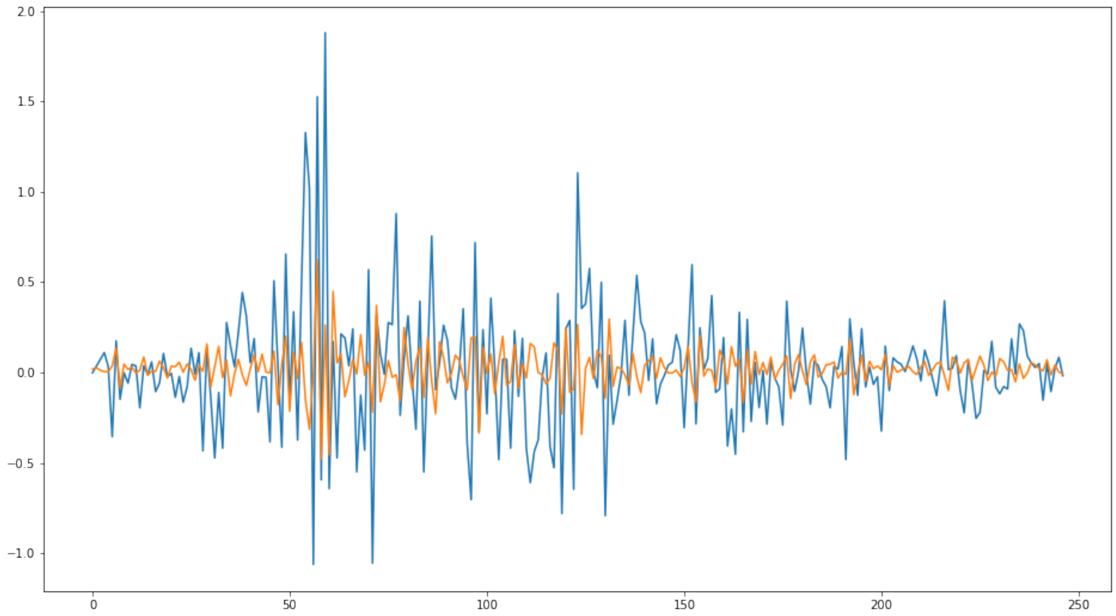
```
(0, 0) 192.80665023518415
(0, 1) 183.66824715354807
(0, 2) 172.94785550694132
(0, 3) 177.43172539946778
(0, 4) 176.220582043445
(1, 0) 177.03889527748385
(1, 1) 178.20279149496992
(1, 2) 174.233164451611
(1, 3) 183.79958962648743
(1, 4) 181.11737921199477
(2, 0) 174.96301675832774
(2, 1) 176.13645538606107
(2, 2) error
(2, 3) error
(2, 4) 183.7705592781964
(3, 0) 176.99509645067036
(3, 1) 181.04908655782322
(3, 2) 179.81491809176617
(3, 3) error
(3, 4) 188.9567476588225
(4, 0) 180.41268832291257
(4, 1) 185.8138812408177
(4, 2) 183.78146079894725
(4, 3) 189.2879747654793
(4, 4) 187.5124519252313
```

```
In [56]: best_score,best_cfg
```

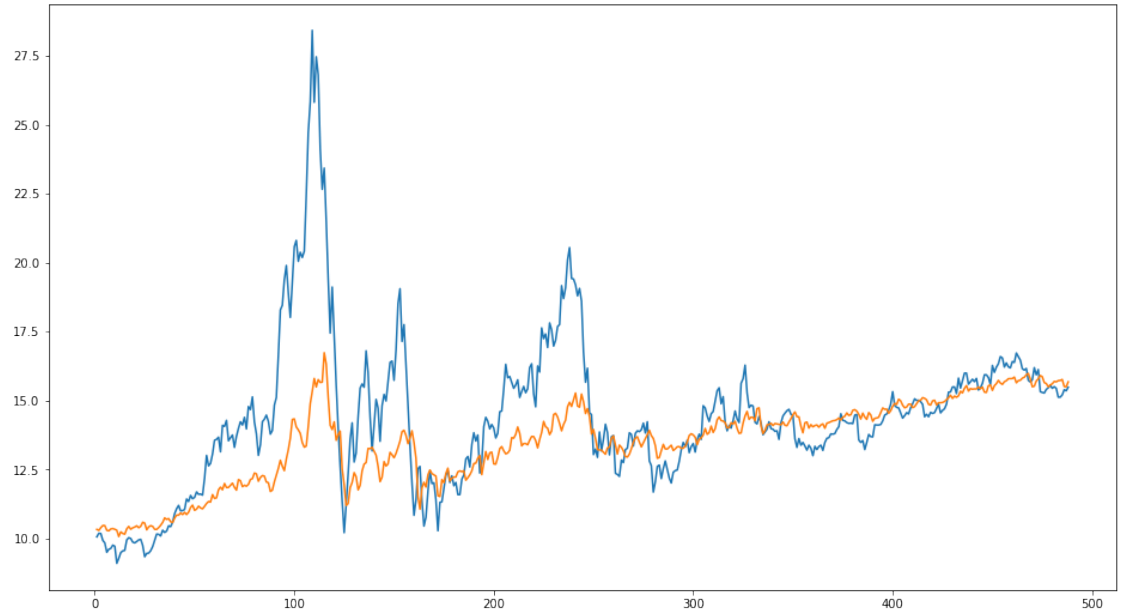
```
Out[56]: (172.94785550694132, (0, 2))
```

```
In [57]: model, model_result = fitModel(coffes[2],best_cfg)
         dp2 = forcastInModel(model_result)
         dpo2,dpci2 = forcastOutModel(model_result,steps[2])
         plt.figure(figsize=(16,9))
         sns.lineplot(list(range(len(coffes[2]))),coffes[2])
         sns.lineplot(list(range(len(dp2))),dp2)
```

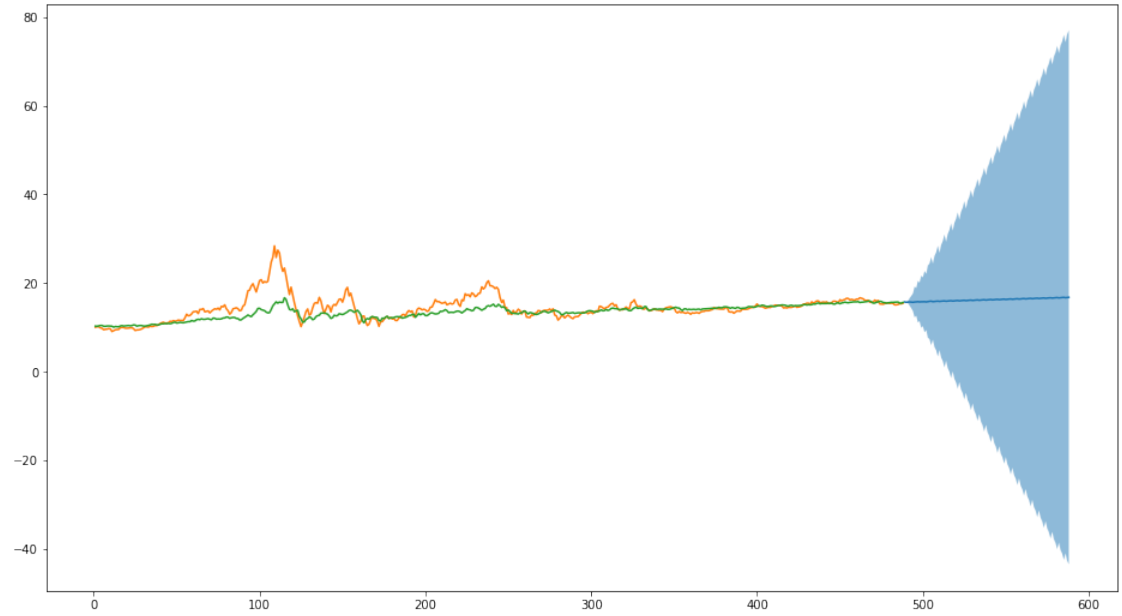
```
Out[57]: <matplotlib.axes._subplots.AxesSubplot at 0x1c3dcf38e80>
```



```
In [58]: coeffnewp = [np.append(apr[0],appr[0][-steps[0]:].values),np.append(dp1,dpo1),np.append(d
         coeffnewu = [np.append(apr[0],apur[0][-steps[0]:].values),np.append(dp1,dpci1[:,0]),np.ap
         coeffnewd = [np.append(apr[0],apdr[0][-steps[0]:].values),np.append(dp1,dpci1[:,1]),np.ap
         dcp = recoverWavelet(coeffnewp)
         dcu = recoverWavelet(coeffnewu)
         dcd = recoverWavelet(coeffnewd)

In [60]: plt.figure(figsize=(16,9))
         sns.lineplot(x,y)
         sns.lineplot(x,dcp[:-100])

Out[60]: <matplotlib.axes._subplots.AxesSubplot at 0x1c3def92940>
```



```
In [61]: plt.figure(figsize=(16,9))
         sns.lineplot(list(range(489,489+step)),dcp[-step:])
         sns.lineplot(x,y)
         sns.lineplot(x,dcp[:-step])
         plt.fill_between(list(range(489,489+step)),dcu[-step:],dcd[-step:],alpha=0.5)

Out[61]: <matplotlib.collections.PolyCollection at 0x1c3decb39e8>
```


80
60
40
20
0
−20
−40
0
100
200
300
400
500
600