In [44]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime as dt
import statsmodels.formula.api as smf
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.holtwinters import SimpleExpSmoothing
from statsmodels.tsa.holtwinters import Holt
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from statsmodels.tsa.stattools import acf,pacf
from statsmodels.tsa.arima_model import ARIMA

import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
data = pd.read_excel('Airlines+Data.xlsx')
data.head(10)
```

Out[3]:

	Month	Passengers
0	1995-01-01	112
1	1995-02-01	118
2	1995-03-01	132
3	1995-04-01	129
4	1995-05-01	121
5	1995-06-01	135
6	1995-07-01	148
7	1995-08-01	148
8	1995-09-01	136
9	1995-10-01	119

In [5]:
```
data.shape
```

Out[5]:
```
(96, 2)
```

In [6]:
```
data.isna().sum()
```

Out[6]:
```
Month         0
Passengers    0
dtype: int64
```

In [7]:
```
data.dtypes
```

Out[7]:
```
Month         datetime64[ns]
Passengers             int64
dtype: object
```

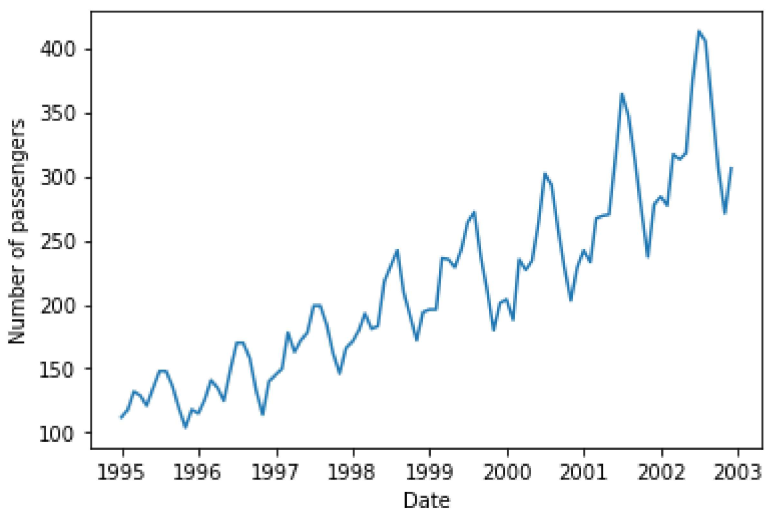
In [8]:
```
data['Month'] = pd.to_datetime(data['Month'],infer_datetime_format=True)
data = data.set_index(['Month'])
```

In [9]:
```
data.head()
```

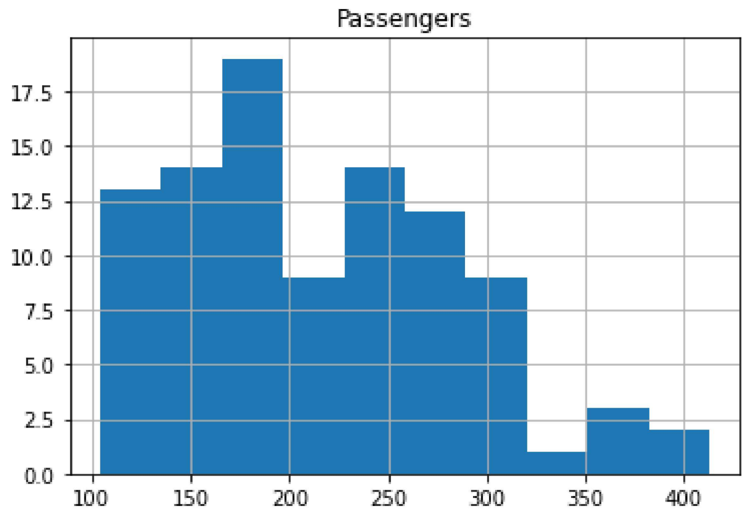
Out[9]:

Month	Passengers
1995-01-01	112
1995-02-01	118
1995-03-01	132
1995-04-01	129
1995-05-01	121

In [11]:
```
plt.plot(data)
plt.xlabel('Date')
plt.ylabel('Number of passengers')
plt.show()
```



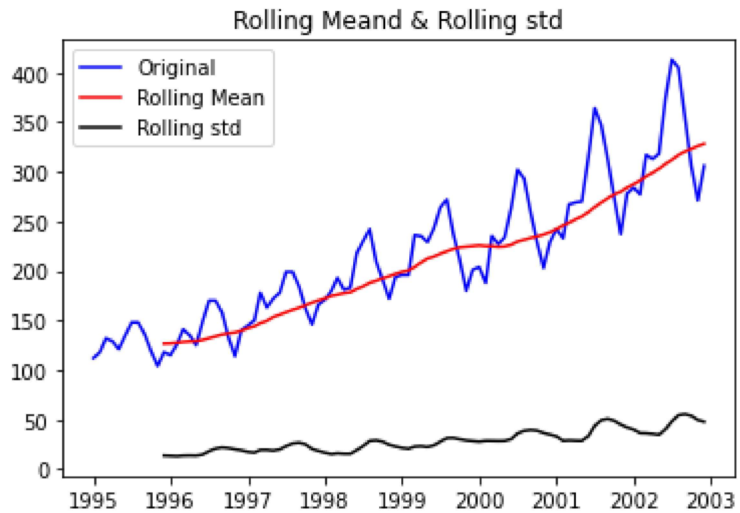
In [14]:
```
data.hist()
plt.show()
```



In [19]:
```
rolmean = data.rolling(window=12).mean()

rolstd = data.rolling(window=12).std()
print(rolmean,rolstd)
```

```
            Passengers
Month
1995-01-01         NaN
1995-02-01         NaN
1995-03-01         NaN
1995-04-01         NaN
1995-05-01         NaN
...                ...
2002-08-01  316.833333
2002-09-01  320.416667
2002-10-01  323.083333
2002-11-01  325.916667
2002-12-01  328.250000

[96 rows x 1 columns]             Passengers
Month
1995-01-01         NaN
1995-02-01         NaN
1995-03-01         NaN
1995-04-01         NaN
1995-05-01         NaN
...                ...
2002-08-01   54.530781
2002-09-01   55.586883
2002-10-01   53.899668
2002-11-01   49.692616
2002-12-01   47.861780

[96 rows x 1 columns]
```

In [21]:
```
orig = plt.plot(data,color='blue',label='Original')
mean = plt.plot(rolmean, color='red',label='Rolling Mean')
std = plt.plot(rolstd, color = 'black',label = 'Rolling std')
plt.legend(loc='best')
plt.title('Rolling Meand & Rolling std')
plt.show(block=False)
```

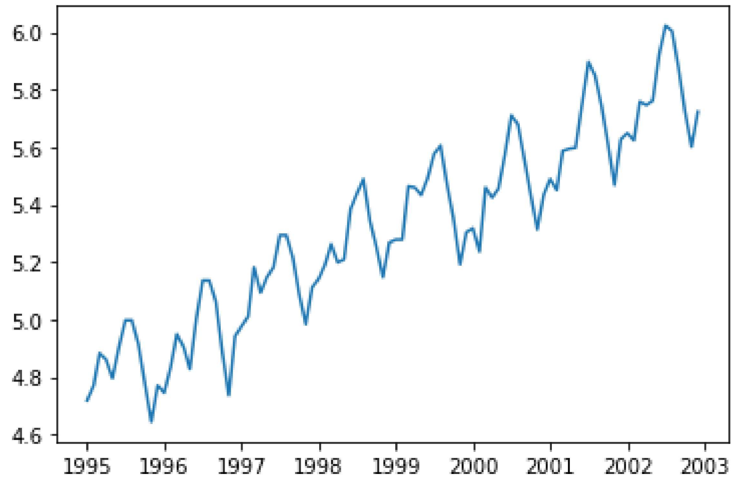


In [22]:
```
from statsmodels.tsa.stattools import adfuller

print('Results of Dickey fuller test:')
dftest = adfuller(data['Passengers'], autolag='AIC')

dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#Lags Used',
for key,value in dftest[4].items():
    dfoutput['Critical Value (%s)'%key]= value

print(dfoutput)
```

```
Results of Dickey fuller test:
Test Statistic                  1.340248
p-value                         0.996825
#Lags Used                     12.000000
Number of Observations Used    83.000000
Critical Value (1%)            -3.511712
Critical Value (5%)            -2.897048
Critical Value (10%)           -2.585713
dtype: float64
```

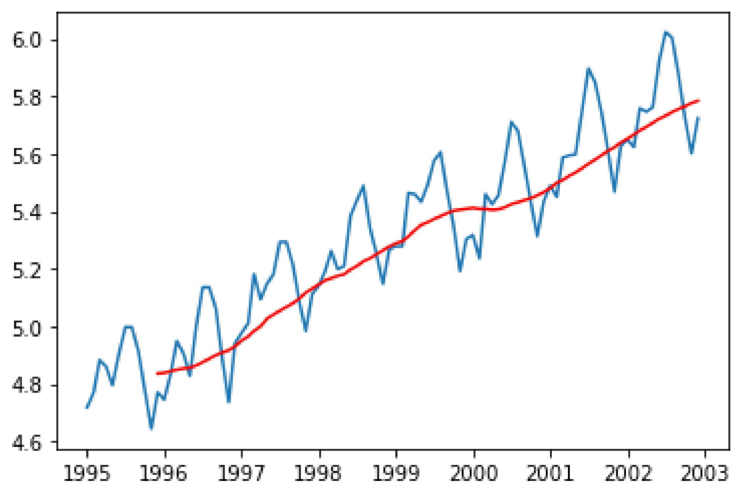
In [24]:
```
data_logScale = np.log(data)
plt.plot(data_logScale)
```

Out[24]: [<matplotlib.lines.Line2D at 0x1f63ae9c550>]



In [25]:
```
movingAverage = data_logScale.rolling(window=12).mean()
movingSTD = data_logScale.rolling(window=12).std()
plt.plot(data_logScale)
plt.plot(movingAverage, color='red')
```

Out[25]: [<matplotlib.lines.Line2D at 0x1f63a43cf70>]



In [27]:
```
datasetLogScaleMinusMovingAverage = data_logScale - movingAverage
datasetLogScaleMinusMovingAverage.head(12)

datasetLogScaleMinusMovingAverage.dropna(inplace=True)
```

In [28]:
```
datasetLogScaleMinusMovingAverage.head()
```

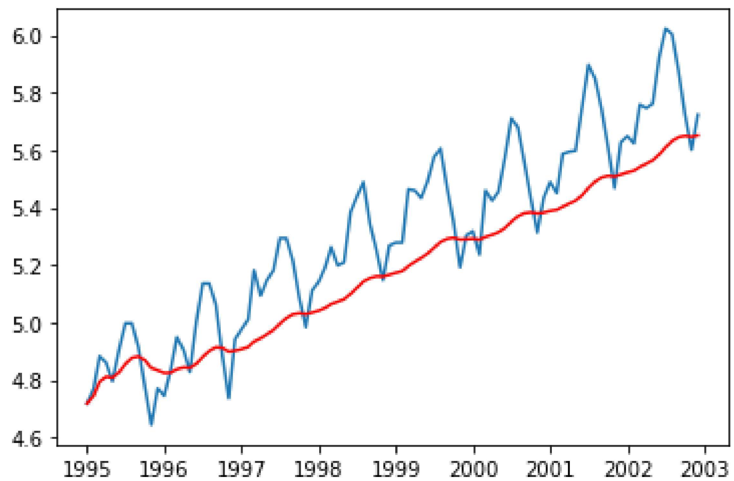
Out[28]:

Month	Passengers
1995-12-01	-0.065494
1996-01-01	-0.093449
1996-02-01	-0.007566
1996-03-01	0.099416
1996-04-01	0.052142

In [29]:
```
def test_stationary(timeseries):

    #Determining rolling statistics
    movingAverage = timeseries.rolling(windows=12).mean()
    movingSTD = timeseries.rolling(window=12).std()

    #plot rolling statistics
    orig = plt.plot(timeseries,color='blue',label='Original')
    mean = plt.plot(movingAverage,color='red',label='Rolling Mean')
    std = plt.plot(movingSTD,color='black',label='Rolling STD')
    plt.legend(loc='best')
    plt.title('Rolling Mean & Standard Deviation')
    plt.show(block=False)

    #Perform Dickey-Fuller test
    print('Results of Dickey fuller test:')
    dftest = adfuller(timeseries['Passengers'], autolag='AIC')
    dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#Lags Us
    for key,value in dftest[4].items():
        dfoutput['Critical Value (%s)'%key]= value

    print(dfoutput)
```

In [32]:
```
exponentialDecayWeightedAverage =data_logScale.ewm(halflife=12, min_periods=0, ac
plt.plot(data_logScale)
plt.plot(exponentialDecayWeightedAverage, color='red')
```

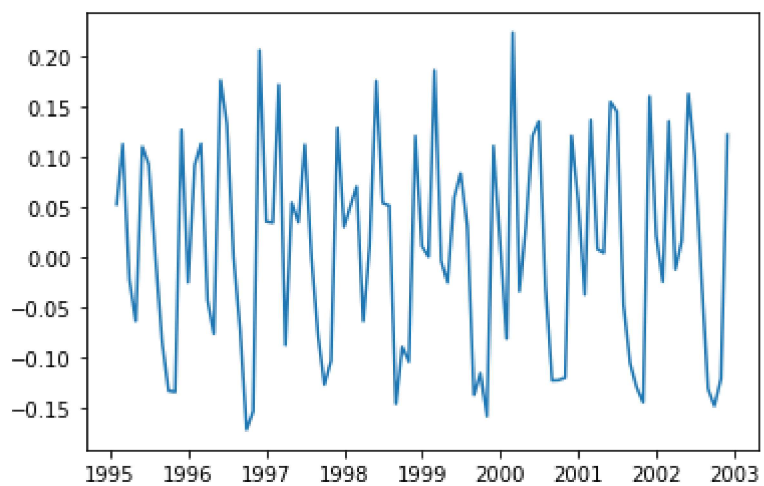
Out[32]: [<matplotlib.lines.Line2D at 0x1f63ab49a90>]



In [33]:
```
eMinusMovingExponentDecayAverage = data_logScale - exponentialDecayWeightedAverag
```

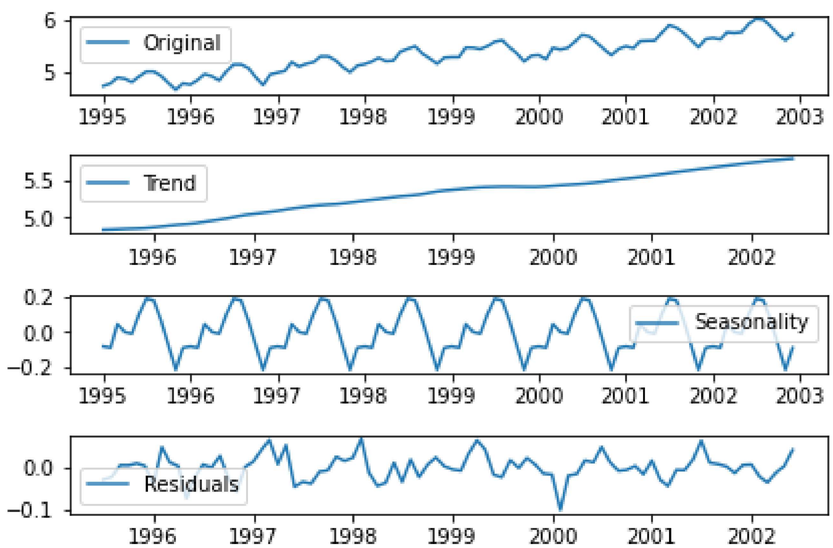
In [36]:
```
datasetLogDiffShifting = data_logScale - data_logScale.shift()
plt.plot(datasetLogDiffShifting)
```

Out[36]: [<matplotlib.lines.Line2D at 0x1f63514a730>]



In [37]:
```
datasetLogDiffShifting.dropna(inplace=True)
```

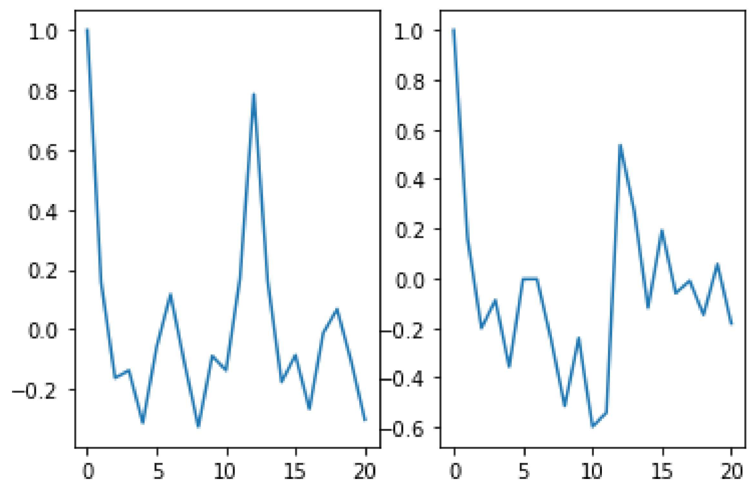
In [39]:
```
decomposition = seasonal_decompose(data_logScale)

trend = decomposition.trend
seasonal= decomposition.seasonal
residual = decomposition.resid

plt.subplot(411)
plt.plot(data_logScale,label='Original')
plt.legend(loc='best')
plt.subplot(412)
plt.plot(trend,label='Trend')
plt.legend(loc='best')
plt.subplot(413)
plt.plot(seasonal,label='Seasonality')
plt.legend(loc='best')
plt.subplot(414)
plt.plot(residual,label='Residuals')
plt.legend(loc='best')
plt.tight_layout()

decompositionLogData = residual
decompositionLogData.dropna(inplace = True)
```



In [40]:
```
decompositionLogData = residual
decompositionLogData.dropna(inplace=True)
```

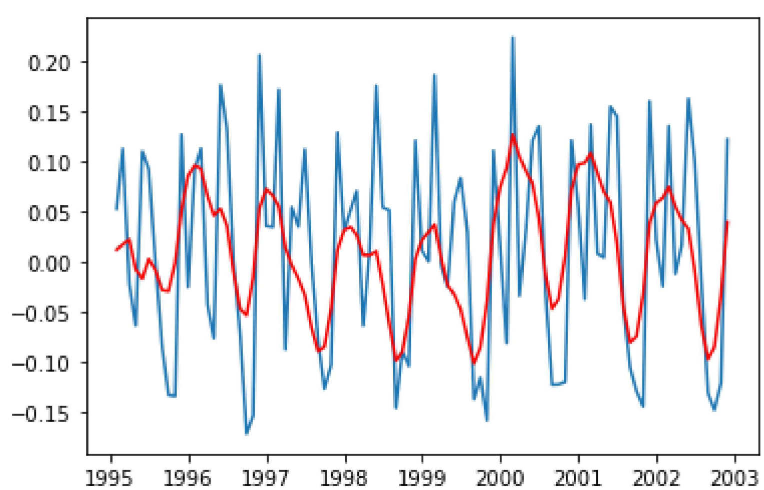
In [43]:
```
lag_acf = acf(datasetLogDiffShifting,nlags=20)
lag_pacf = pacf(datasetLogDiffShifting,nlags=20,method='ols')

#plot ACF
plt.subplot(121)
plt.plot(lag_acf)


#plot PACF
plt.subplot(122)
plt.plot(lag_pacf)
```

Out[43]:
```
[<matplotlib.lines.Line2D at 0x1f63c885850>]
```



In [46]:
```
model = ARIMA(data_logScale,order=(2,1,2))
results_AR = model.fit(disp=-1)
plt.plot(datasetLogDiffShifting)
plt.plot(results_AR.fittedvalues, color='red')
print('plotting AR Model')
```

```
plotting AR Model
```



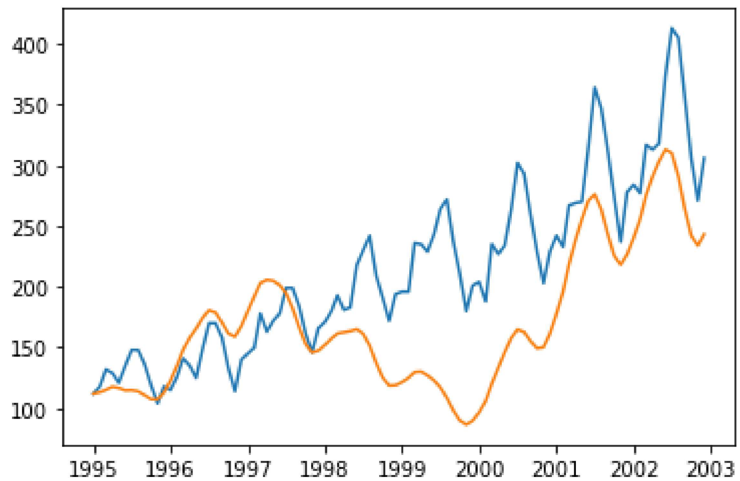
In [47]:
```
predictions_ARIMA_diff = pd.Series(results_AR.fittedvalues, copy=True)
```

In [48]:
```
predictions_ARIMA_diff_cumsum = predictions_ARIMA_diff.cumsum()
```

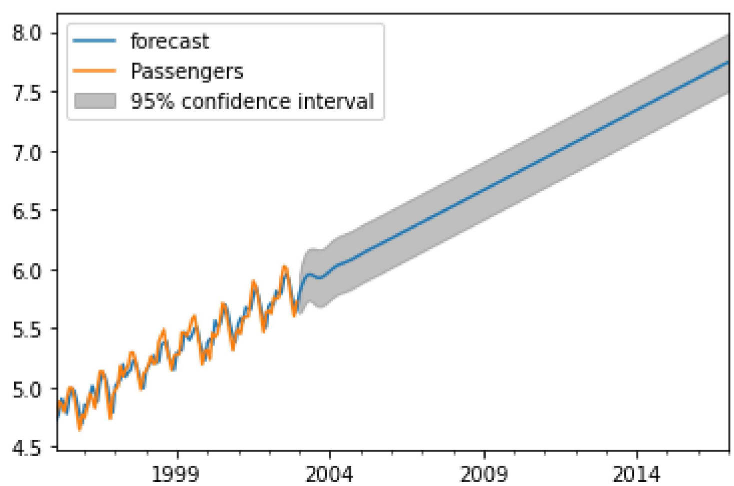
In [49]:
```
IA_log = pd.Series(data_logScale['Passengers'].iloc[0], index =data_logScale.index
IA_log = predictions_ARIMA_log.add(predictions_ARIMA_diff_cumsum,fill_value=0)
```

In [51]:
```
predictions_ARIMA = np.exp(predictions_ARIMA_log)
plt.plot(data)
plt.plot(predictions_ARIMA)
```

Out[51]: [<matplotlib.lines.Line2D at 0x1f63e0e3160>]



In [52]:
```
results_AR.plot_predict(1,264)
x=results_AR.forecast(steps=120)
```



In [53]:
```
x[1]
```

Out[53]:
```
array([0.08322475, 0.10432532, 0.10878105, 0.10878453, 0.11128772,
       0.11614942, 0.12023477, 0.12203416, 0.12227852, 0.122341  ,
       0.12285478, 0.12358269, 0.12409375, 0.12426872, 0.12427673,
       0.12430624, 0.12440361, 0.12450991, 0.12457006, 0.12458411,
       0.12458415, 0.12459315, 0.12461084, 0.12462623, 0.12463324,
       0.12463425, 0.12463445, 0.12463626, 0.12463885, 0.12464065,
       0.12464122, 0.12464124, 0.1246414 , 0.12464185, 0.12464234,
       0.12464263, 0.12464271, 0.12464271, 0.12464273, 0.12464277,
       0.1246428 , 0.12464282, 0.12464282, 0.12464282, 0.12464284,
       0.12464287, 0.12464289, 0.1246429 , 0.1246429 , 0.1246429 ,
       0.1246429 , 0.1246429 , 0.1246429 , 0.12464291, 0.12464291,
       0.12464291, 0.12464292, 0.12464292, 0.12464293, 0.12464293,
       0.12464293, 0.12464294, 0.12464294, 0.12464294, 0.12464294,
       0.12464295, 0.12464295, 0.12464295, 0.12464296, 0.12464296,
       0.12464296, 0.12464296, 0.12464297, 0.12464297, 0.12464297,
       0.12464298, 0.12464298, 0.12464298, 0.12464299, 0.12464299,
       0.12464299, 0.12464299, 0.124643  , 0.124643  , 0.124643  ,
       0.12464301, 0.12464301, 0.12464301, 0.12464301, 0.12464302,
       0.12464302, 0.12464302, 0.12464303, 0.12464303, 0.12464303,
       0.12464304, 0.12464304, 0.12464304, 0.12464304, 0.12464305,
       0.12464305, 0.12464305, 0.12464306, 0.12464306, 0.12464306,
       0.12464307, 0.12464307, 0.12464307, 0.12464307, 0.12464308,
       0.12464308, 0.12464308, 0.12464309, 0.12464309, 0.12464309,
       0.12464309, 0.1246431 , 0.1246431 , 0.1246431 , 0.12464311])
```

In [54]:
```
len(x[1])
```

Out[54]:
```
120
```

In [ ]: