University of New Brunswick

Time Series Analysis
(EE 6563)

Assignment #1

Professor:
Erik Scheme
Electrical and Computer
Engineering

Author:
Saeed Kazemi
(3713280)

February 6, 2021

1. **For each dataset, visualize the raw signals, identify any trends, seasonality, and/or other components, and try to remove them. Remember that you are not limited to the tools shown in the tutorial and should explore the various concepts discussed in class. You do not need to explain the theory behind the approaches, but you should provide justification for their use, and discussion of their results.**

   Figure 1 and 2 indicate the raw signal of both data sets. Furthermore, figure 3 to 5 illustrate a period of two datasets.

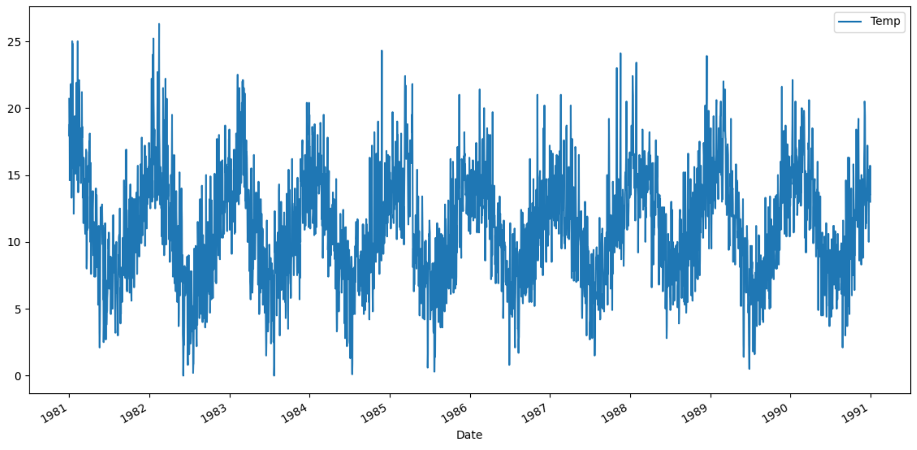

Figure 1: Visualizing the raw signals of the first dataset.

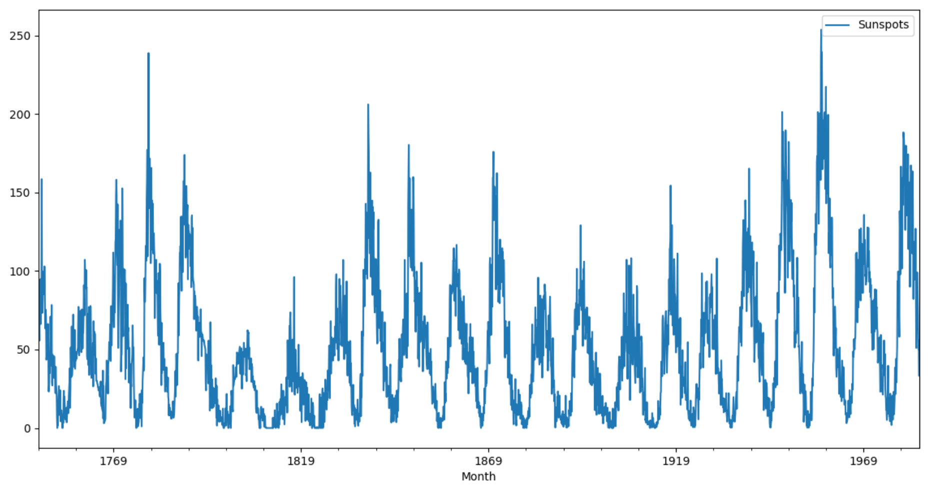

Figure 2: Visualizing the raw signals of the second dataset.

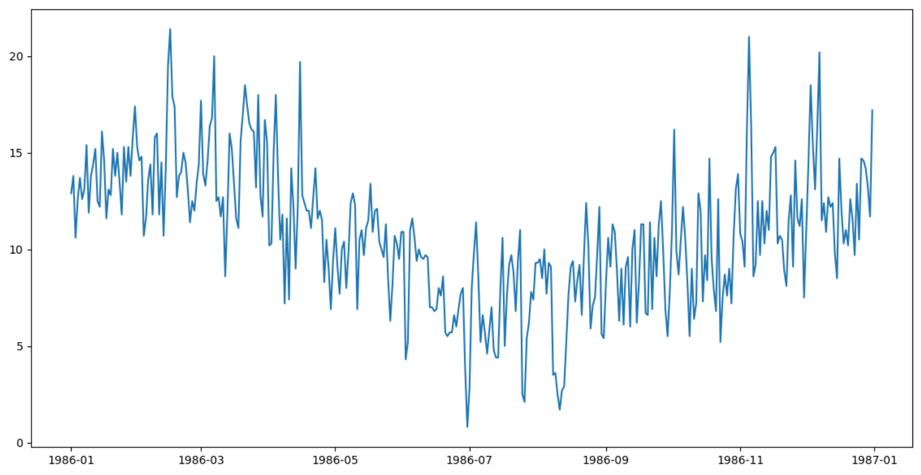

Figure 3: Visualizing the raw signals of the first dataset in 1986.

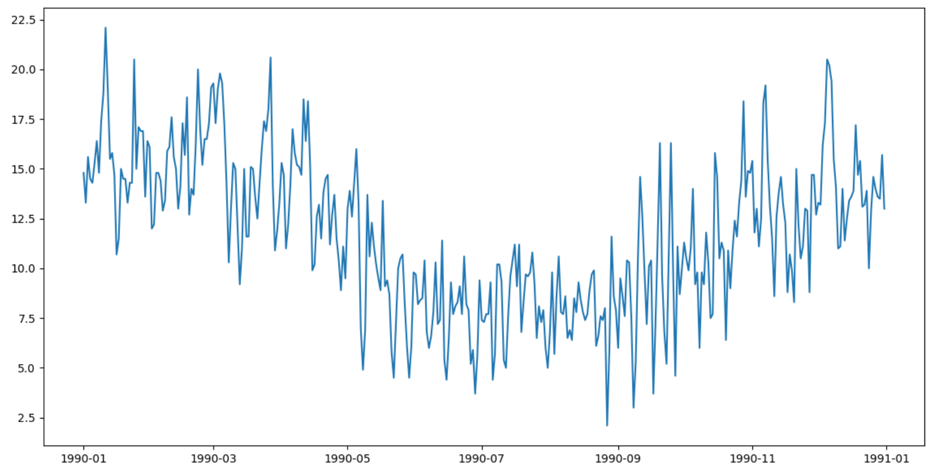

Figure 4: Visualizing the raw signals of the first dataset in 1990.

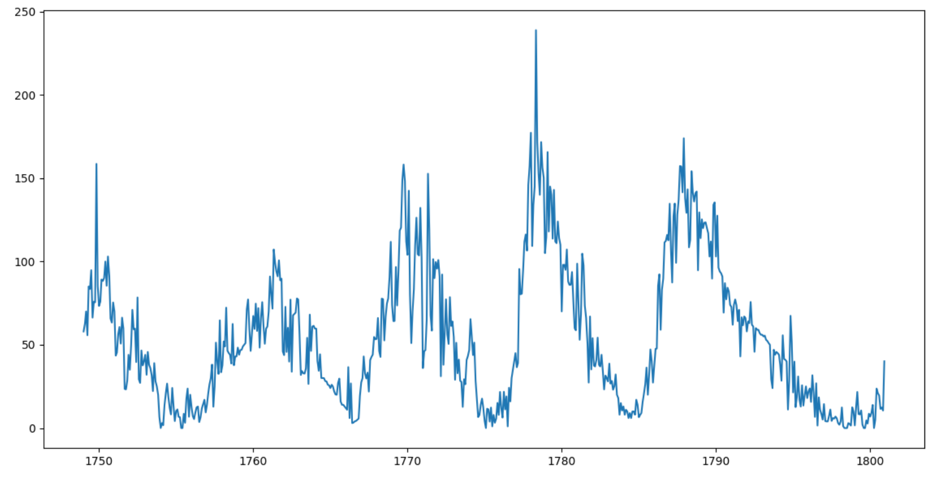

Figure 5: Visualizing the raw signals of the second dataset in 1990.

For intuition, Table 1 to 4 show the top raw of two datasets along with the summary of two time series datasets.

Table 1: The top row of the raw signal of the first dataset (daily temp).

	Temp
Date	
1981-01-01	20.7
1981-01-02	17.9
1981-01-03	18.8
1981-01-04	14.6
1981-01-05	15.8

Table 2: The description of the first dataset.

	Temp
count	3650.000000
mean	11.177753
std	4.071837
min	0.000000
25%	8.300000
50%	11.000000
75%	14.000000
max	26.300000

Table 3: The top row of the raw signal of the second dataset.

	Sunspots
Month	
1749-01-01	58.0
1749-02-01	62.6
1749-03-01	70.0
1749-04-01	55.7
1749-05-01	85.0

Table 4: The description of the second dataset.

	Sunspots
count	2820.000000
mean	51.265957
std	43.448971
min	0.000000
25%	15.700000
50%	42.000000
75%	74.925000
max	253.800000

For decomposing the data the below methods were used:

(a) *Seasonal_decompose (Figure 6 and 7)*

(b) *STL (Figure 8 and 9)*

(c) *Linear Regression method (Figure 10 and 11)*

(d) *Difference method (Figure 12 and 13)*

(e) *Fitting a polynomial (Figure 14 and 15)*

(f) *Moving Average window (Figure 16 and 17)*

Some mentioned methods are used only for extracting only one component while others like STL and Seasonal_decompose provided all three components. Table 5 compares these methods together.

Also there are two model for reconstruction of time series, Additive Model and Multiplicative Model. In this assignment the additive model was used because Multiplicative is not appropriate for zero and negative values.

Table 5: Comparing the implemented methods.

	Trend component	Seasonal component	Residual component
Seasonal_decompose	✓	✓	✓
STL	✓	✓	✓
Linear Regression	✓	-	-
Difference	✓	✓	✓
Fitting polynomial	✓	✓	✓
Moving average Function	✓	-	-

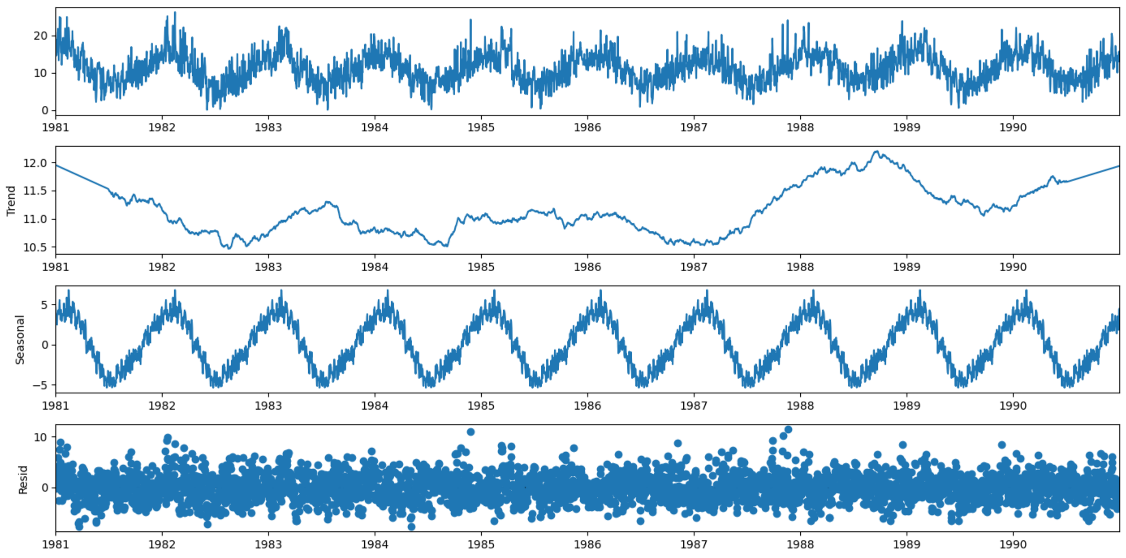

Figure 6: Decomposition of the first dataset by seasonal_decompose method

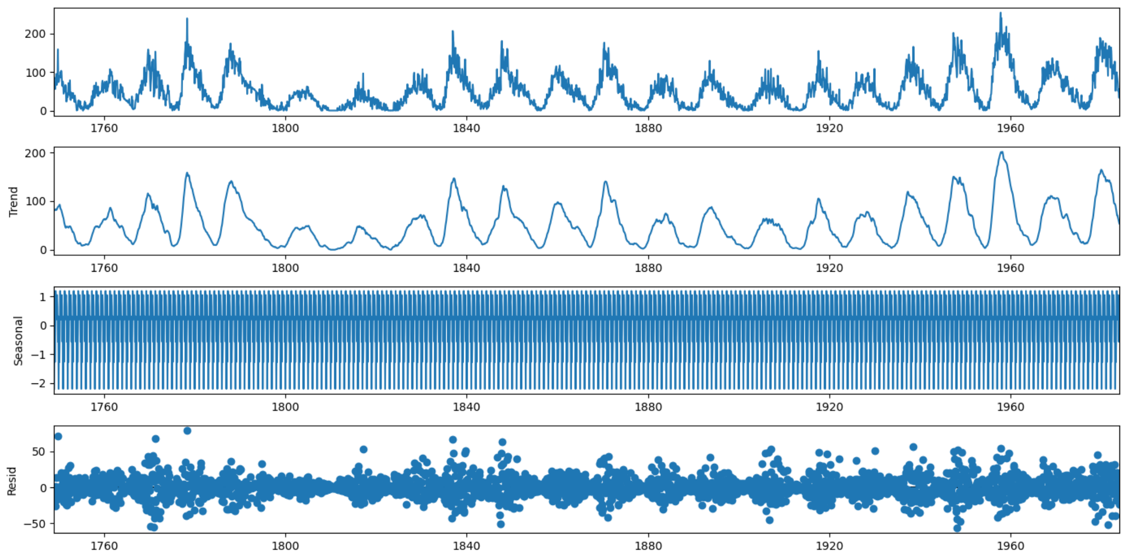

Figure 7: Decomposition of the second dataset by seasonal_decompose method

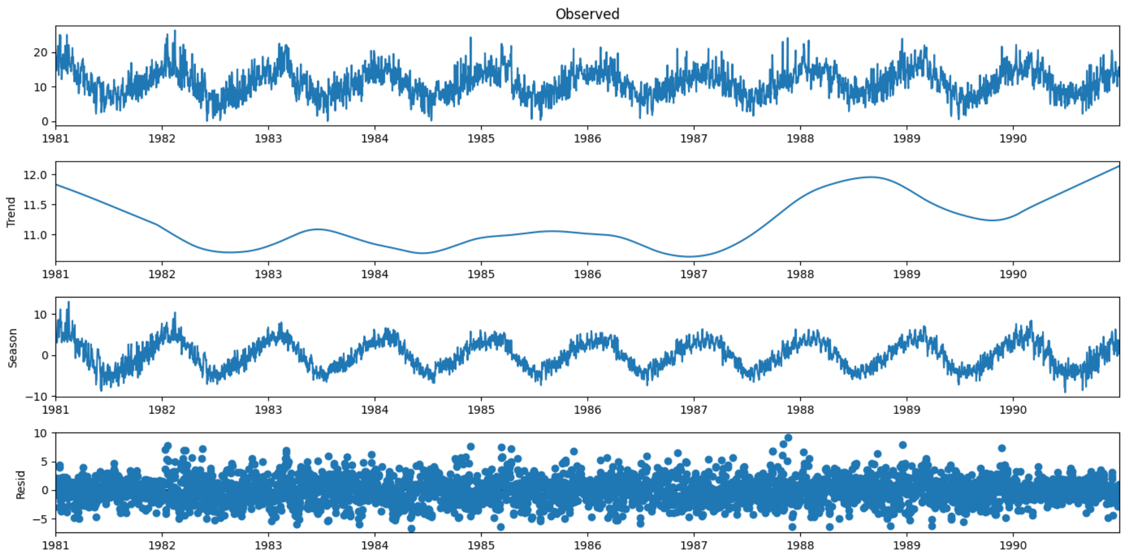

Figure 8: Decomposition of the first dataset by STL method

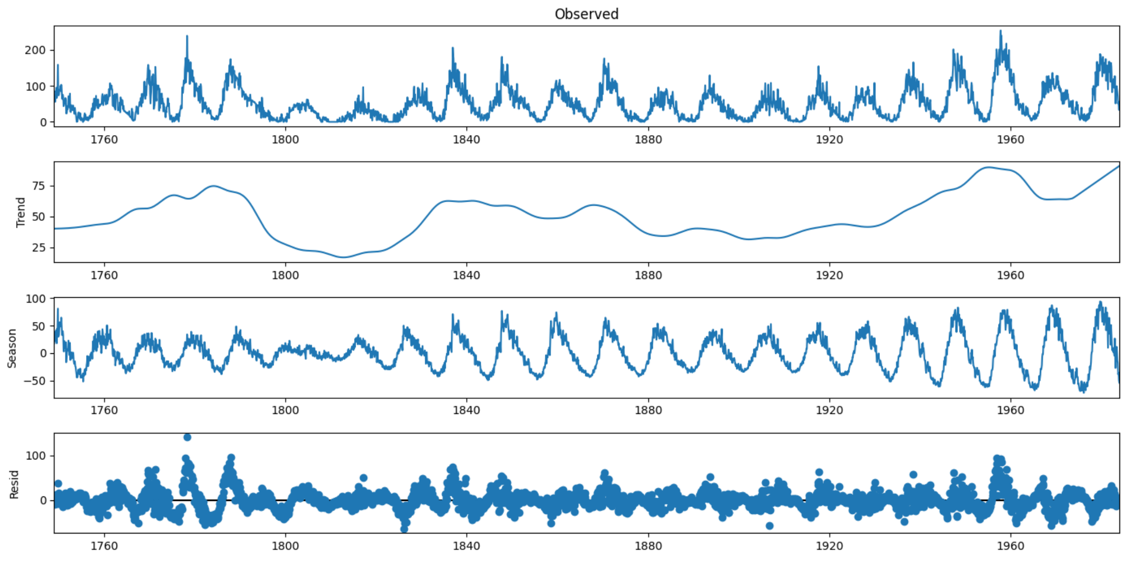

Figure 9: Decomposition of the second dataset by STL method

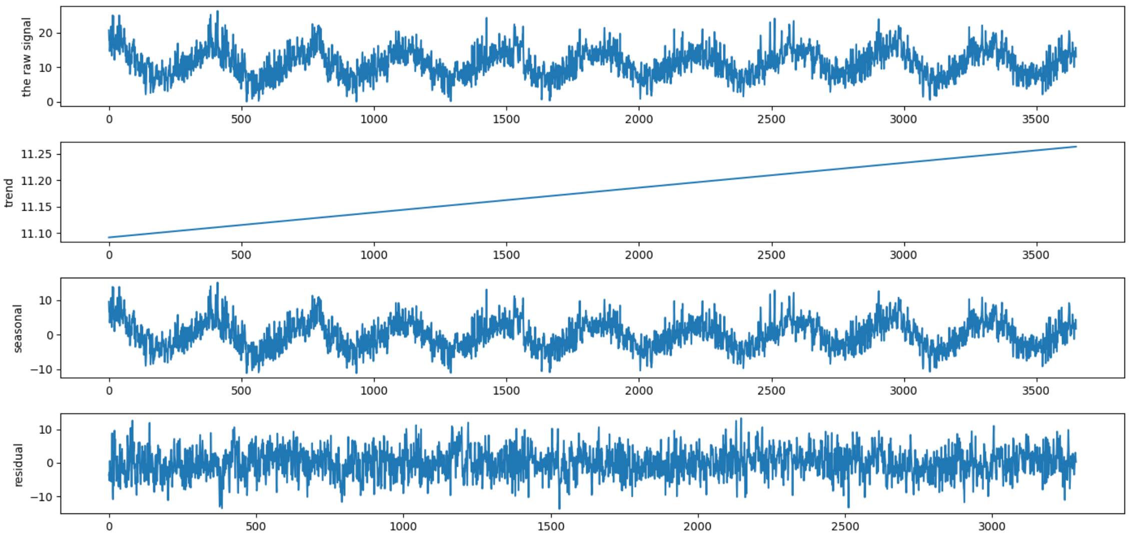

Figure 10: Decomposition of the first dataset by LinearRegression and difference method.

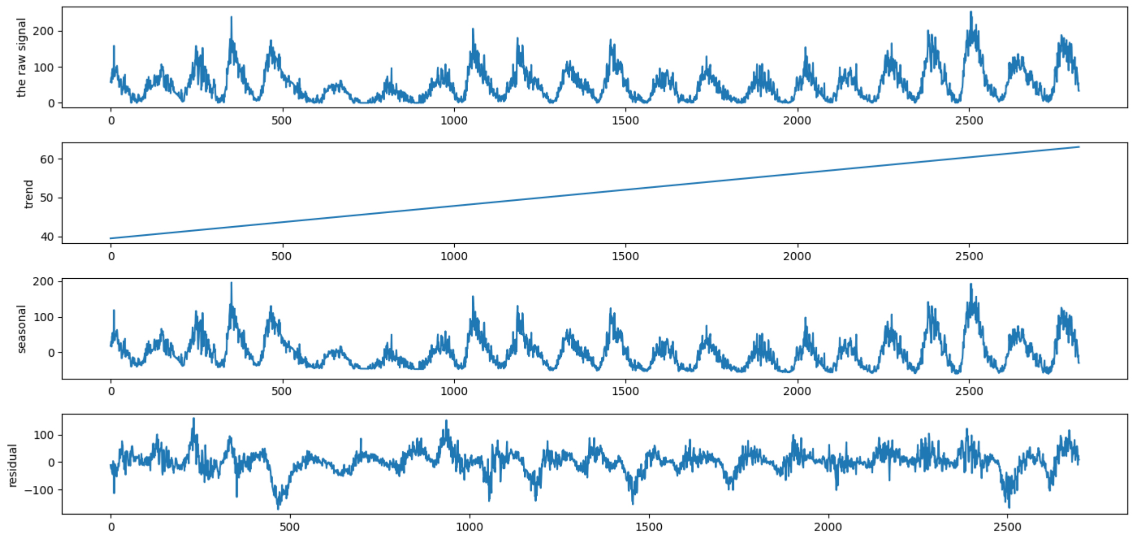

Figure 11: Decomposition of the second dataset by LinearRegression and difference method.

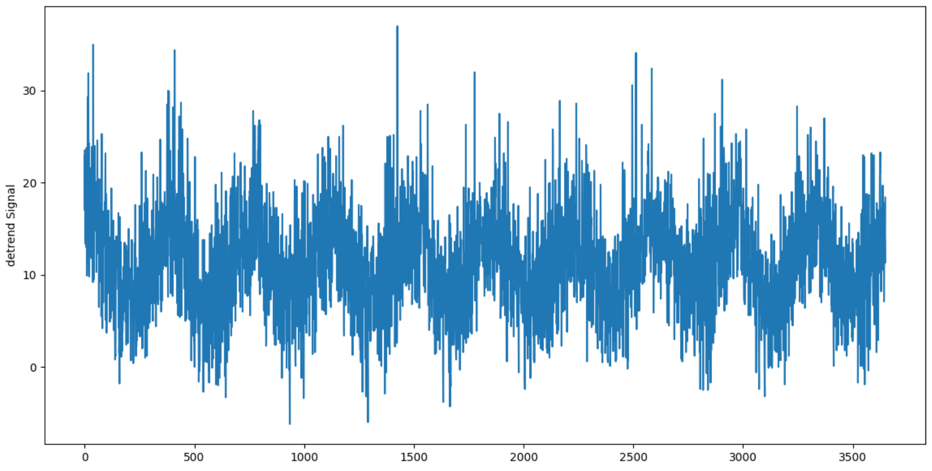

Figure 12: Detrending of the first dataset by difference method.

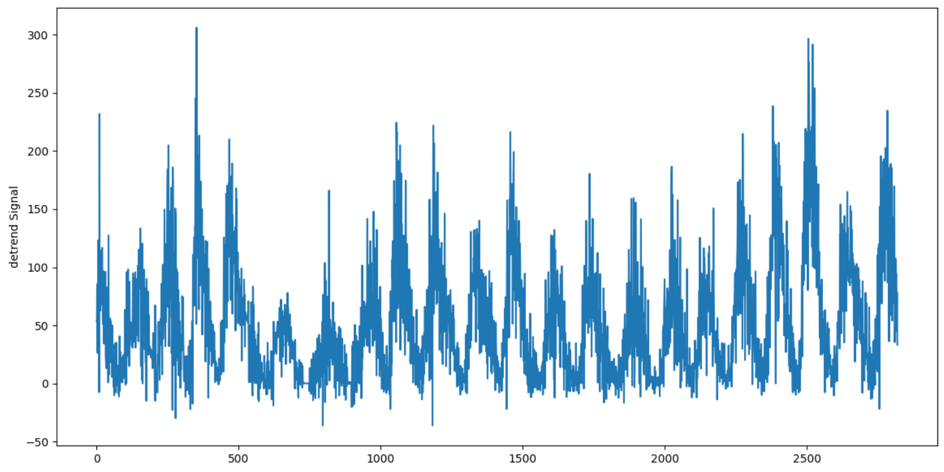

Figure 13: Detrending of the second dataset by difference method.

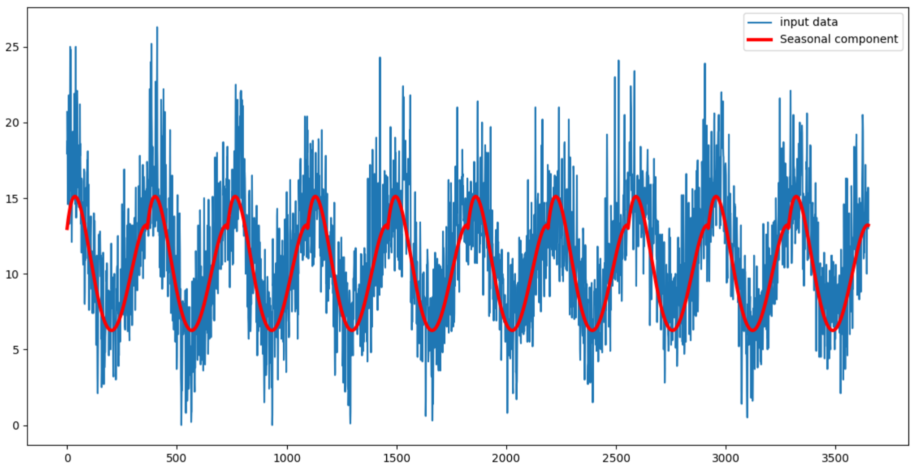

Figure 14: Seasonal component of the first dataset by fitting a polynomial.

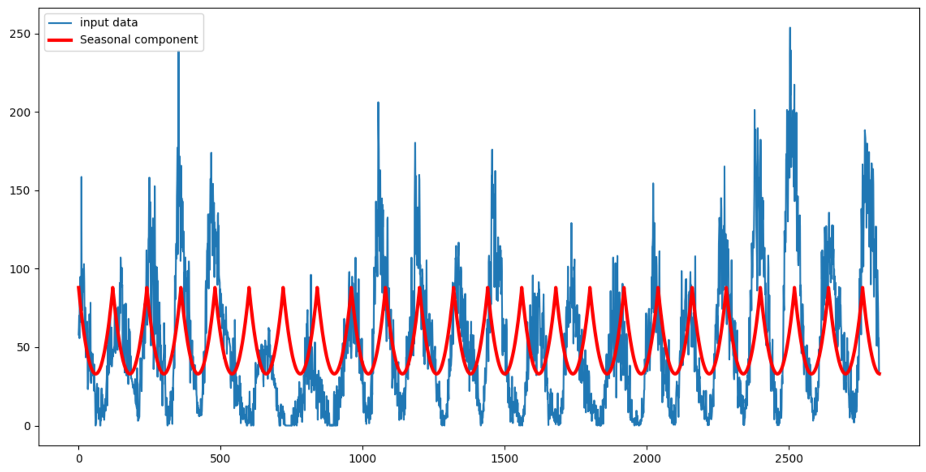

Figure 15: Seasonal component of the second dataset by fitting a polynomial.

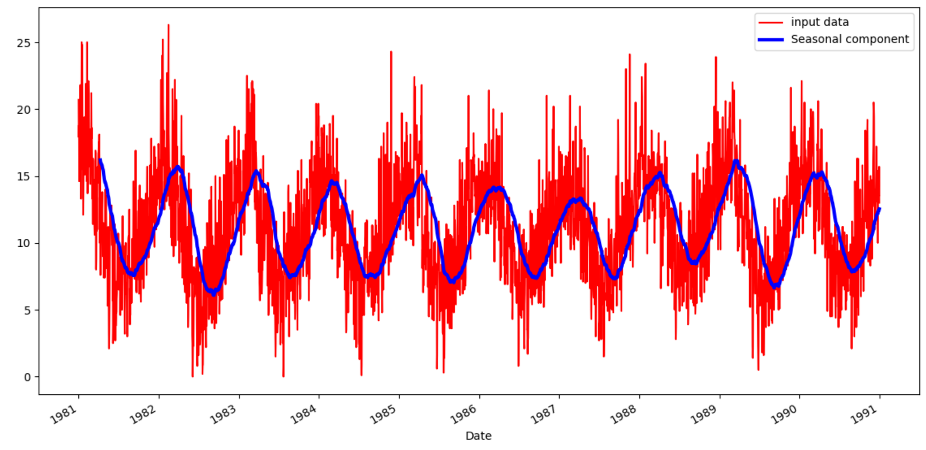

Figure 16: Seasonal component of the first dataset by Moving Average.

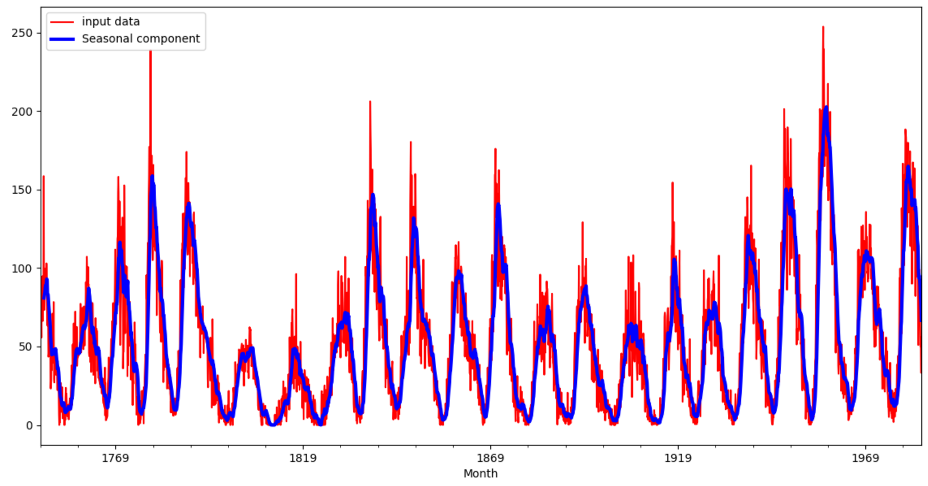

Figure 17: Seasonal component of the second dataset by Moving Average.

2. **For each dataset, examine the stationarity of the residuals using the ACF and PACF functions, Lag Plots, and/or other approaches. Show your results and provide commentary about your observations.**

   Augmented Dickey-Fuller test (ADF test) shows that the time series is a non-stationary or stationary. If the p-value is very less than significance level of 0.05 and the ADF test statistic is lower than one of the critical values, then the time series is a stationary. For example, table 6 shows a stationary time series.

Table 6: The result of the ADF test on the first dataset.

	0
ADF Statistic	-24.697687
p-value	0.000000
#Lags Used	3.000000
Number of Observations Used	3646.000000
Critical Value (1%)	-3.432145
Critical Value (5%)	-2.862333
Critical Value (10%)	-2.567192

Table 7: The result of the ADF test on the second dataset.

	0
ADF Statistic	-1.128610e+01
p-value	1.416821e-20
#Lags Used	2.700000e+01
Number of Observations Used	2.792000e+03
Critical Value (1%)	-3.432694e+00
Critical Value (5%)	-2.862576e+00
Critical Value (10%)	-2.567321e+00

Kwiatkowski-Phillips-Schmidt-Shin test (KPSS test) is another test for checking the stationarity of a time series. If the p-value is very less than significance level of 0.05, then the time series is not a stationary. Table 8 and 9 show the result of this test.

Table 8: The result of the KPSS test on the first dataset.

	0
KPSS Statistic	0.019544
p-value	0.100000
Lags Used	21.000000
Critical Value (10%)	0.347000
Critical Value (5%)	0.463000
Critical Value (2.5%)	0.574000
Critical Value (1%)	0.739000

Table 9: The result of the KPSS test on the second dataset.

	0
KPSS Statistic	0.007897
p-value	0.100000
Lags Used	29.000000
Critical Value (10%)	0.347000
Critical Value (5%)	0.463000
Critical Value (2.5%)	0.574000
Critical Value (1%)	0.739000

We can conclude that the series is stationary or not based on the both result of KPSS test and ADF test [1]. Table 11 shows possible outcomes of applying these two tests.

Table 10: The combination of the result of the KPSS test and ADF test.

KPSS test	KDF test	The combination result
non-stationary	non-stationary	The series is non-stationary.
stationary	non-stationary	Use detrending to make series stationary.
non-stationary	stationary	Use differencing to make series stationary.
stationary	stationary	The series is stationary.

AutoCorrelation function (ACF) and Partial AutoCorrelation function (PACF) plots allow you to determine the time series at zero hoe much correlation has with other lags. Figure 18 and 19 indicate these two plot for our datasets.

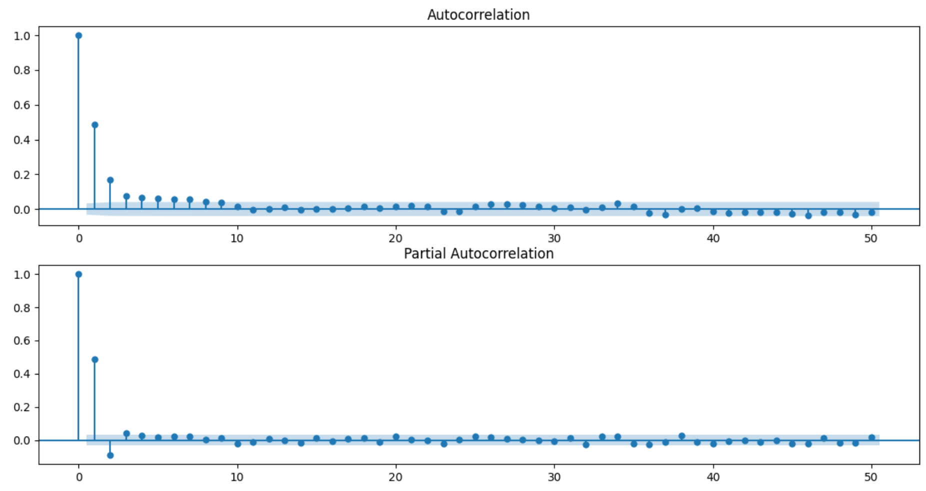

Figure 18: A plot of the ACF and PACF of the first dataset.

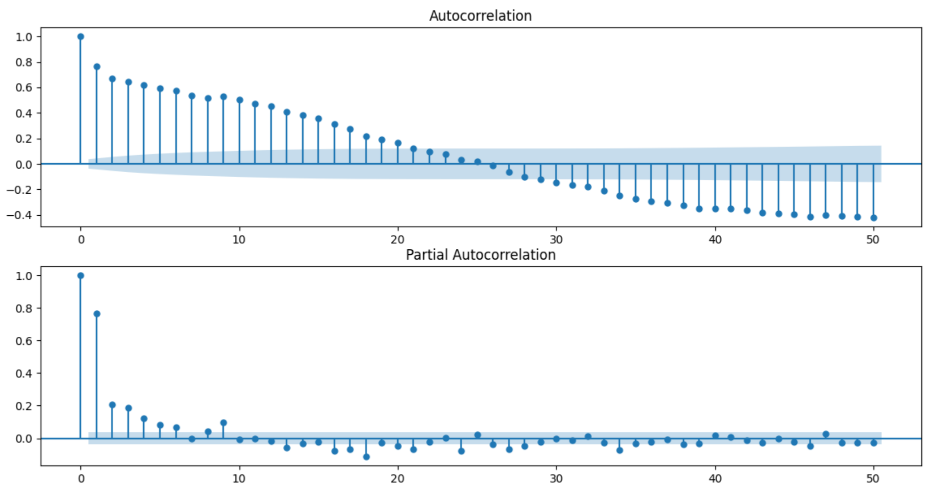

Figure 19: A plot of the ACF and PACF of the second dataset.

These two plots are used to find the q and p for the ARIMA model. For example, if ACF decays towards zero, and PACF have only q significant value then our time series is a AR(q) process.

Table 11: The combination of the result of the KPSS test and ADF test.

KPSS test	KDF test	The combination result
non-stationary	non-stationary	The series is non-stationary.
stationary	non-stationary	Use detrending to make series stationary.
non-stationary	stationary	Use differencing to make series stationary.
stationary	stationary	The series is stationary.

The below figures (figure 20 and 21) show the lag plot of the two datasets. In each figure, there are four lag plots. As these plots illustrate, both datasets are linear, therefore our time series are a AR process.

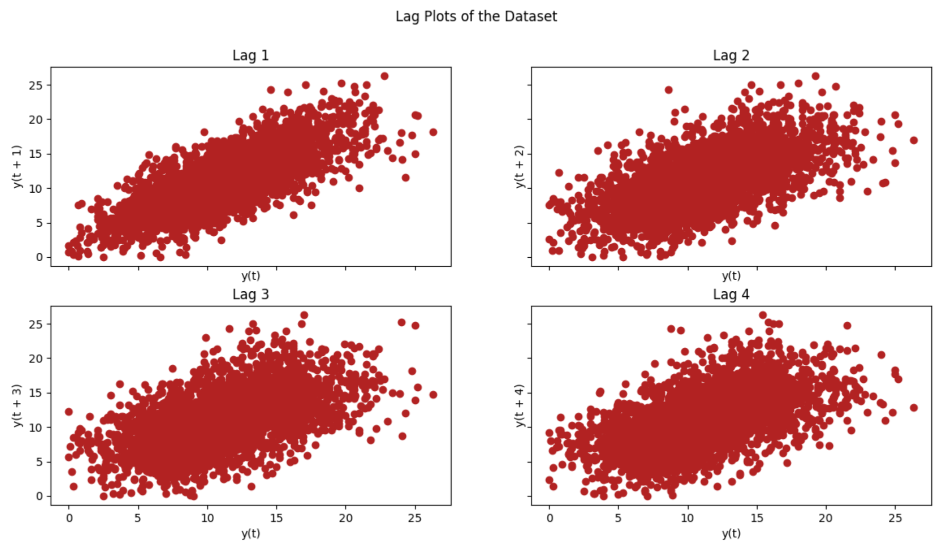

Figure 20: A different lag plot of the first dataset.

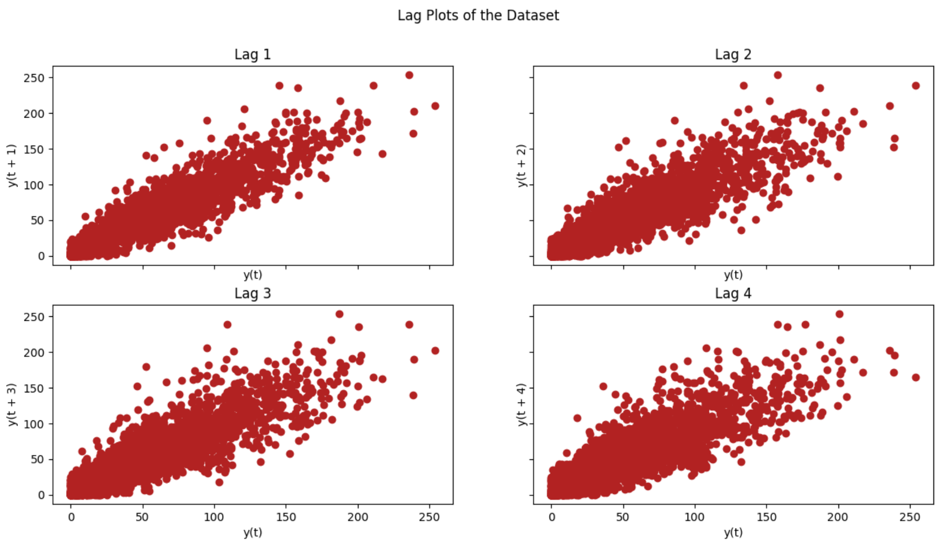

Figure 21: A different lag plot of the second dataset.

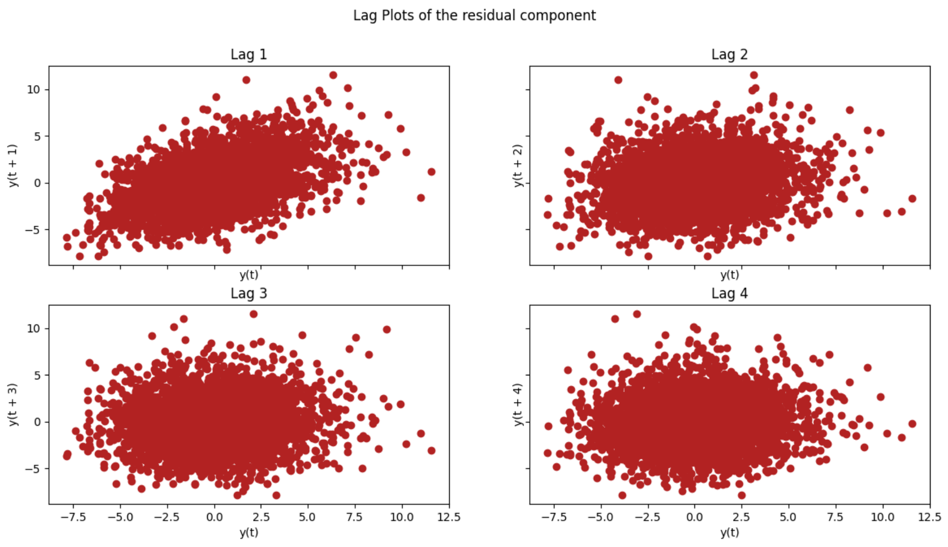

Figure 22: A different lag plot of residual component of the first dataset.

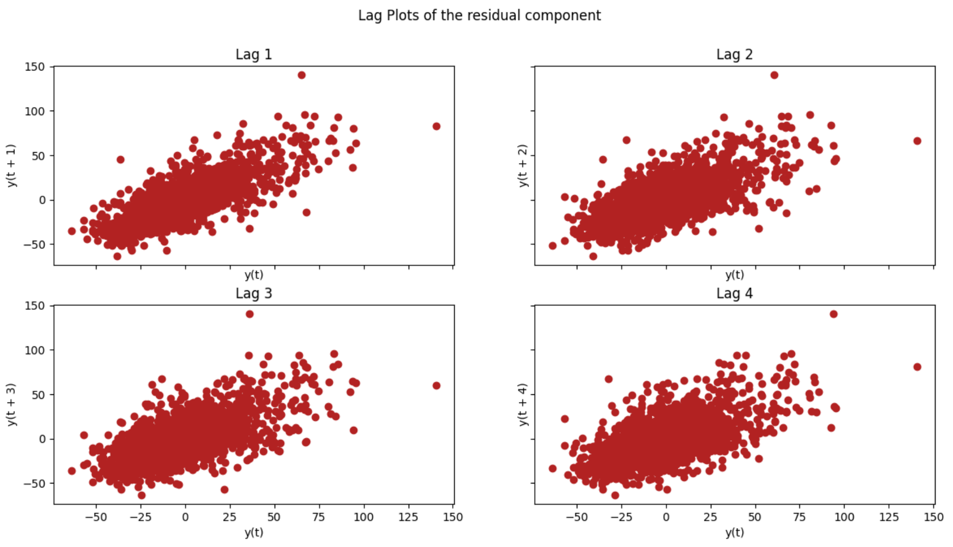

Figure 23: A different lag plot of residual component of the second dataset.

3. **Try modeling the residuals as an AR process. Use the tools at your disposal to decide on an appropriate order and analyse the results. What is the impact of selecting different orders on the remaining residuals?**

4. **Summarize your findings and observations briefly in a final discussion. Submit both the developed code and your document to the Assignment 1 folder on D2L.**

   All files were uploaded on my GitHub repository []. The below table, shows the important paths of the project. Beside the codes are available in Appendix

The Important Paths:	
./EE6563/Dataset/Ass1	Assignment datasets
./EE6563/code/Ass1/sun.py	Assignment code
./EE6563/code/Ass1/Temp.py	Assignment code
./EE6563/manuscript/src/figures	Assignment figures
./EE6563/manuscript/src/tables	Assignment table
./EE6563/manuscript/src/Ass1.tex	Assignment document

References

[1] “Stationarity and detrending (ADF/KPSS) — statsmodels.”

1 Appendix (codes)

1.1 The script of Lab-1

```
%% Lab 1: Discrete-Time Frequency
%
% Author: Maryhelen Stevenson
%
%% Explanation of the function ct_dt(.)
%
% The file ct_dt.m was supplied for use in this lab; a listing ...
    of the file
% is included in Appendix 1. The file defines a matlab function
% ct_dt(A,f0,PHI,fs,nc,ifig) which can be used to plot nc cycles
% of a continuous-time cosine with amplitude A, frequency f0, ...
    and phase
% PHI.  It also superimposes the values of the discrete-time ...
    sinusoid that
% would result from sampling the continuous-time sinusoid at a ...
    rate of fs
% samples per second.  The function returns a vector of time
% instances at which the values of the continuous-time sinusoid were
% evaluated to generate the continuous-time plot.
%
%
% An example to illustrate the usage of ct_dt(.) follows:
%
close all
A = 2; % amplitude of continuous-time cosine
f0 = 0.5; % frequency (in units of Hz.) of continuous-time cosine
PHI = -pi/4; % phase (in units of radians) of continuous-time cosine
fs = 10; % sampling rate to be used (units of samples/second)
nc = 5; % number of cycles of continuous-time cosine to be plotted
ifig = 1; % optional figure number to use in the title of the figure
% the function returns the time vector used to plot the c.t. signal
t = ct_dt(A,f0,PHI,fs,nc,ifig);

%%
% _Discussion of Figure 1_
% In accordance with the usage of ct_dt, we note that Figure 1,
% contains 5 cycles of the continuous-time cosine, $$x_a(t)$,
% where $$x_a(t)=2\cos(2\pi(0.5)t - \pi/4)$.  It also ...
    superimposes the
% discrete-time sinusoid x[n] = xa(n/10).  The values of n are ...
    not shown
% but could be added in by hand.  Note that t=0 corresponds to ...
    n=0; whereas
```

```
% t=1 corresponds to n=10.
% etc.
%
%% Exercise 1
%
% Let x_a(t) = 3 sin(2 pi 50 t) = 3 cos (2 pi 50 t - pi/2)
%
% Define x[n] = x_a(n/fs)
%
% a) Figure 2 shows a plot of xa(t) and x[n] for the case when ... fs=200
% samples/second.  It was produced using the code below.
%
%  include the necessary code
close all
A = 3;
f0 = 50;
PHI = -pi/2;
fs = 200;
nc = 6;
ifig = 2;


t = ct_dt(A,f0,PHI,fs,nc,ifig);


%%
% _Discussion of Figure 2_
%
% include discussion here.  Your discussion should include ... answers to all
% questions posed in the lab manual.  Please use complete ... sentences.  Keep
% in mind that a reader should not have to have a copy of the ... lab manual to
% make sense of your discussion.
%%
% b) Figure 3 shows a plot of xa(t) and x[n] for the case when ... fs=120
% samples/second. It was produced using the code below.
%
%  include the necessary code
fs = 120;
ifig = 3;


t = ct_dt(A,f0,PHI,fs,nc,ifig);
%%
% _Discussion of Figure 3_
```

```
%
% include discussion here
%%
% c) Figure 4 shows a plot of xa(t) and x[n] for the case when fs=40
% samples/second. It was produced using the code below.
%
%  include the necessary code
fs = 40;
ifig = 4;


t = ct_dt(A,f0,PHI,fs,nc,ifig);
%%
% _Discussion of Figure 4_
%
% include discussion here.
%
%%
% d) ...
%
%% Exercise 2
%
%

close all
A = 6;
f0 = 50;
PHI = -pi/3;
fs = 100;
nc = 2;
ifig = 21;


t = ct_dt(A,f0,PHI,fs,nc,ifig);


A = 3;
f0 = 50;
PHI = 0;
fs = 100;
nc = 2;
ifig = 22;


t = ct_dt(A,f0,PHI,fs,nc,ifig);


A = 6;
f0 = 50;
```

```
PHI = -pi/3;
fs = 100;
nc = 2;
ifig = 23;


t = ct_dt(A,f0,PHI,fs,nc,ifig);
y = 3*cos(2*pi*50*t);
plot(t,y,'*')
legend('x(t)','x[n]','y(t)')
%% Exercise 3
%
%



close all
A = 1;
f0 = 50;
PHI = -pi/2;
fs = 80;
nc = 5;
ifig = 31;


t = ct_dt(A,f0,PHI,fs,nc,ifig);


y = cos(2*pi*30*t + pi/2);

plot(t,y,'*')
legend('x(t)','x[n]','y(t)')


%% Exercise 4
%
%
close all
A = 1;
f0 = 60;
PHI = 0;
fs = 50;
nc = 12;
ifig = 41;


t = ct_dt(A,f0,PHI,fs,nc,ifig);

y = cos(2*pi*10*t);
```

```

plot(t,y,'*')
legend('x(t)','x[n]','y(t)')

%% Appendix 1: Listing of the file ct_dt.m
%
%     Please include a listing of the function here, complete ...
    with any
%     modifications that you may have made.
%
%     function t = ct_dt(A,f0,PHI,fs,nc, ifig)
%     %A         amplitude of cosine;
%     %f0   CT frequency of cosine (cycles/sec);
%     %PHI  phase of cosine (radians);
%     %fs        sampling frequency (samples/sec.)
%     %nc   number of CT cycles to be displayed
%     %ifig   optional Figure number to use in the title of the plot
%
%     ...
```

1.2 The function of ct_dt

```
function t = ct_dt(A,f0,PHI,fs,nc, ifig)
%A      amplitude of cosine;
%f0     CT frequency of cosine (cycles/sec);
%PHI    phase of cosine (radians);
%fs     sampling frequency (samples/sec.)
%nc     number of CT cycles to be displayed
%ifig   option Figure number to use in the title of the plot
%t      time vector used to plot CT cosine
if (nargin < 5 | nargin > 6)
    error('in call to ct_dt: there should be 5 arguments')
end
if (A < 0 )
    error(['in call to ct_dt: Amplitude of cosine, A,' ...
        'should not be negative'])
end
if (fs<0)
    error(['in call to ct_dt: the sampling frequency,'...
           ' fs,should be positive'])
end
if (nc<0)
    error('in call to ct_dt: nc should be positive')
end
if (exist('ifig'))
    pFig = ['Fig. ', num2str(ifig),':  '];
```

```
else
    pFig = [''];
end


figure, clf
Ts=1/fs; %time between samples
Tp=1/abs(f0);  %period of CT cosine (sec/cycle)
F0 = f0/fs; %DT frequency (cycles/sample)
%DT plot will display samples n=0 to n=nmax
nmax = nc/abs(F0);
%CT plot will display t=0 to t=tmax
tmax = nmax * Ts;
% define t vector for CT plots to:
%  have a length greater than or equal to 200
%  with every kth element corresponding to a sampling instant
k = ceil(200/nmax);
t=0:Ts/k:tmax;
xa = A*cos(2*pi*f0*t + PHI);
plot(t,xa);
hold on
n=0:nmax;
nTs = n*Ts;
xn = A*cos(2*pi*F0*n + PHI);
stem(nTs,xn);

p0=['x_a(t)=A cos(2\pi f_0 t), '];
p1=[' A='];
p2=[', f0='];
p3=[', PHI='];
p4=[', fs='];
p5=[', nc='];
p6=[', User='];
name=getenv('USER'); % gets the users login id
title( [pFig p0 p1 num2str(A) p2 num2str(f0) p3 num2str(PHI) ...
         p4 num2str(fs) p5 num2str(nc) p6 name ] )
xlabel('t (seconds), n')
```