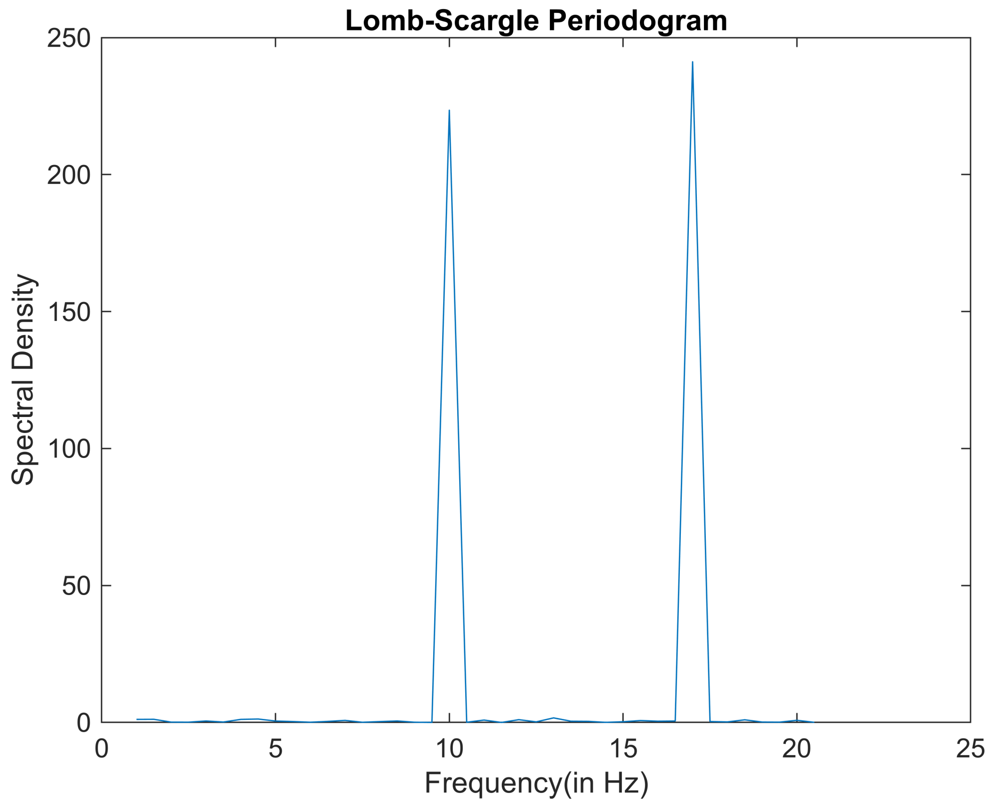
Problem 2: Lomb-Scargle Periodogram

```
%Problem 2 part a
T=linspace(0,10,500);
T=T';
W1=sin(2*pi*10*T);
W2=sin(2*pi*17*T);
W_0=W1+W2; %DGP
W_0=W_0-mean(W_0); %mean centred signal
for i=1:50
    rand_index=randi([1,size(W_0,1)]);
    T(rand_index)=[];
    W_0(rand_index)=[];
end
varianceW_0=var(W_0);
SNR=10;
e_var=var(W_0)/SNR;
noise=normrnd(0,sqrt(e_var),[450,1]);
W_meas=W_0+noise;
```

```
frequencies=[1:0.5:20.5];
Y=W_meas;
spec_density=zeros(40,1);
j=1;
for f=frequencies %writing gradient descent for each frequency and calculating
spectral densities
    N=450;
    [theta,X]=gradDes(f,N,T,Y);
    spec_density(j)=norm(X*theta).^2;
    j=j+1;
end
```

```
plot(frequencies,spec_density)
xlabel("Frequency(in Hz)")
ylabel("Spectral Density")
title("Lomb-Scargle Periodogram")
```




```
%computing maximum frequencies obtained
[m,idx]=max(spec_density);
maxf1=frequencies(idx)
```

```
maxf1 = 17
```

```
spec_density(idx)=[];
[m,idx]=max(spec_density);
maxf2=frequencies(idx)
```

```
maxf2 = 10
```

```
f=maxf1;
N=450;
theta1=gradDes(maxf1,450,T,Y)
```

```
theta1 = 2×1
    0.0015
    1.0316
```

```
theta2=gradDes(maxf2,450,T,Y)
```

```
theta2 = 2×1
    0.0153
    0.9947
```

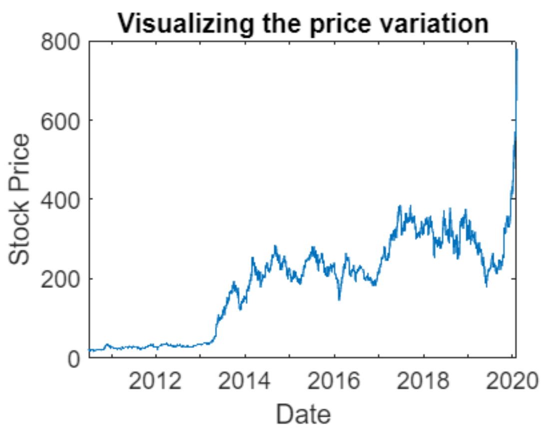
```
function [theta,X]=gradDes(f,N,T,Y)
```

```
    x1=cos(2*pi*f*T);
    x2=sin(2*pi*f*T);
    X=[x1 x2] ;%phi matrix

    thetavec_0=[1;1];

    % Initial guess
    theta = thetavec_0;

    % Learning rate
    alpha = 0.1;

    % Stopping criterion (tolerance for convergence)
    tolerance = 0.001;

    % Maximum number of iterations (to prevent infinite loops in case of poor
convergence)
    max_iterations = 1000;

    % Track cost history
    cost_history = zeros(max_iterations, 1);

    % Gradient descent loop
    for i = 2:max_iterations
      % Calculate prediction
      prediction = X * theta;

      % Calculate cost (mean squared error)
      cost_history(i) =sum((prediction - Y) .^ 2);
      % Calculate gradient
      gradient = X' * (prediction - Y)./N;

      % Update theta using gradient descent
      theta = theta - alpha * gradient;

      % Check stopping criterion (absolute difference in cost)
      if abs(cost_history(i) - cost_history(i-1)) < tolerance
        break;
      end
    end
end
```

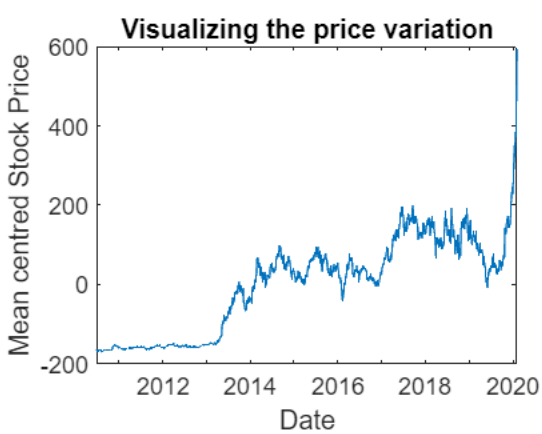
Q 2b) Using the Lomb-Scargle Periodogram to predict stock prices

To get a Lomb-Scargle periodogram, we need to take appropraite frequencies. Since, the data is given for 10 years the frequency can vary anywhere from once every 10 years to daily which is what is assumed below.

```
minFreq = 1 / 3650; % Minimum frequency (cycles per day)
maxFreq = 1;        % Maximum frequency (cycles per day)
frequencies=[minFreq:minFreq:maxFreq];
Y=Close;
Y=Y-mean(Y);
dateTimes = datetime(Date, 'InputFormat', 'dd-MM-yyyy');
plot(dateTimes,Close)
xlabel("Date")
ylabel("Stock Price")
title("Visualizing the price variation")
```



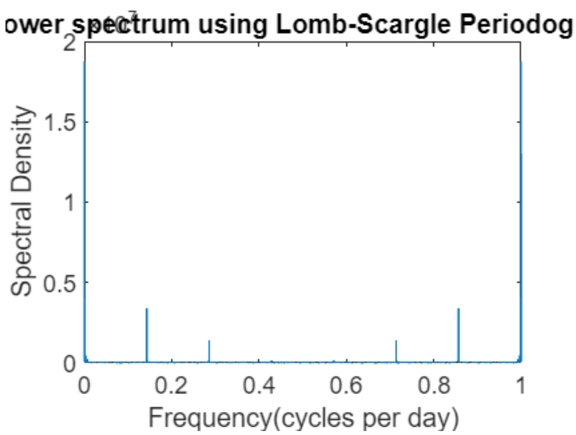

```
plot(dateTimes,Y)
xlabel("Date")
ylabel("Mean centred Stock Price")
title("Visualizing the price variation")
```




```
Y_test=zeros(400,1);
Date_train=Date;
Date_test=zeros(400,1);
Date_test=string(Date_test);
for i=1:400 %randomly collected testing set
```

```
    rand_index=randi([1,size(Date_train,1)]);
    Date_test(i)=Date_train(rand_index);
    Date_train(rand_index)=[];
    Y_test(i)=Y(rand_index);
    Y(rand_index)=[];
end

%training set
dateTimes = datetime(Date_train, 'InputFormat', 'dd-MM-yyyy');
T = datenum(dateTimes);
%test set
dateTimes_test=datetime(Date_test, 'InputFormat', 'dd-MM-yyyy');
T_test=datenum(dateTimes_test);
spec_density=zeros(size(frequencies,2),1);
j=1;
for f=frequencies
    N=2016;
    x1=cos(2*pi*f*T);
    x2=sin(2*pi*f*T);
    X=[x1 x2] ;%phi matrix
    theta=gradDes(f,N,T,Y);
    spec_density(j)=norm(X*theta).^2;
    j=j+1;
end
```

```
plot(frequencies,spec_density)
xlabel("Frequency(cycles per day)")
ylabel("Spectral Density")
title("Power spectrum using Lomb-Scargle Periodogram")
```




```
[pk,f0] = findpeaks(spec_density,frequencies','MinPeakHeight',0.7*1e6)
```

```
pk = 9×1
10^7 ×
    0.0945
    0.3377
    0.1335
    0.0787
```

```
    0.0787
    0.1335
    0.3378
    0.0945
    1.8773
f0 = 9×1
    0.0014
    0.1430
    0.2855
    0.2860
    0.7140
    0.7145
    0.8570
    0.9986
    0.9997
```

```
%pk are the maxima obtained & f0 are the corresponding frequencies
theta_ans=gradDes2(f0,2016,T,Y)
```

```
theta_ans = 18×1
  -10.4114
   -9.1689
    1.9714
    8.3716
    0.9613
   -2.2210
    0.3135
    0.7760
    0.3135
    1.2240
      ⋮
```

```
j=1;
guess=zeros(400,1); %test guesses
guess2=zeros(2016,1); %training guesses
for i=1:size(f0,1)
    guess=guess+theta_ans(j)*cos(2*pi*f0(i)*T_test)
+theta_ans(j+1)*sin(2*pi*f0(i)*T_test);
    guess2=guess2+theta_ans(j)*cos(2*pi*f0(i)*T)+theta_ans(j+1)*sin(2*pi*f0(i)*T);
    j=j+2;
end
```

The errors are defined according to the definitions given: Normalized Mean Square Error and Mean Absolute Percentage Error

```
%evaluating test and train error
%normalized mean square error
MSE_train=sqrt(mean((Y-guess2).^2))/(mean(abs(Y)))
```

```
MSE_train = 0.6578
```

```
MSE_test=sqrt(mean((Y_test-guess).^2))/(mean(abs(Y_test)))
```

```
MSE_test = 0.6575
```

```
%Absolute percentage error
```

```
abs_perc_error_test=mean(abs((Y_test-guess)./Y_test))*100
```

```
abs_perc_error_test = 198.2909
```

```
abs_perc_error_train=mean(abs((Y-guess2)./Y))*100
```

```
abs_perc_error_train = 281.6353
```

The gradDes2 function is to evaluate the coefficients whose sines and cosines summed can be used to predict the stock prices. The frequencies which have highest spectral densities are obtained from the Lomb-Scargle Periodogram.

```
function theta=gradDes2(f0,N,T,Y) %gradient descent to get coefficients
    X=[];
    for i=1:size(f0,1)
        u1=cos(2*pi*f0(i)*T);
        u2=sin(2*pi*f0(i)*T);
        X=[X u1 u2];
    end
    thetavec_0=ones(2*size(f0,1),1);
    % Initial guess
    theta = thetavec_0;

    % Learning rate
    alpha = 0.1;

    % Stopping criterion (tolerance for convergence)
    tolerance = 0.001;

    max_iterations = 1000;

    % Gradient descent loop
    for i = 2:max_iterations
      % Calculate prediction
      prediction = X * theta;
      % Calculate gradient
      gradient = X' * (prediction - Y)./N;
      % Update theta using gradient descent
      theta = theta - alpha * gradient;
    end
end
function theta=gradDes(f,N,T,Y)
    x1=cos(2*pi*f*T);
    x2=sin(2*pi*f*T);
    X=[x1 x2] ;%phi matrix

    thetavec_0=[1;1];

    % Initial guess
    theta = thetavec_0;
```

```
    % Learning rate
    alpha = 0.1;

    % Stopping criterion (tolerance for convergence)
    tolerance = 0.001;

    % Maximum number of iterations (to prevent infinite loops in case of poor
convergence)
    max_iterations = 1000;

    % Track cost history
    cost_history = zeros(max_iterations, 1);

    % Gradient descent loop
    for i = 2:max_iterations
      % Calculate prediction
      prediction = X * theta;

      % Calculate cost (mean squared error)
      cost_history(i) =sum((prediction - Y) .^ 2);
      % Calculate gradient
      gradient = X' * (prediction - Y)./N;

      % Update theta using gradient descent
      theta = theta - alpha * gradient;

      % Check stopping criterion (absolute difference in cost)
      if abs(cost_history(i) - cost_history(i-1)) < tolerance
        break;
      end
    end
end
```

Implementing an ARIMA Model to fit the Tesla Stock Prices

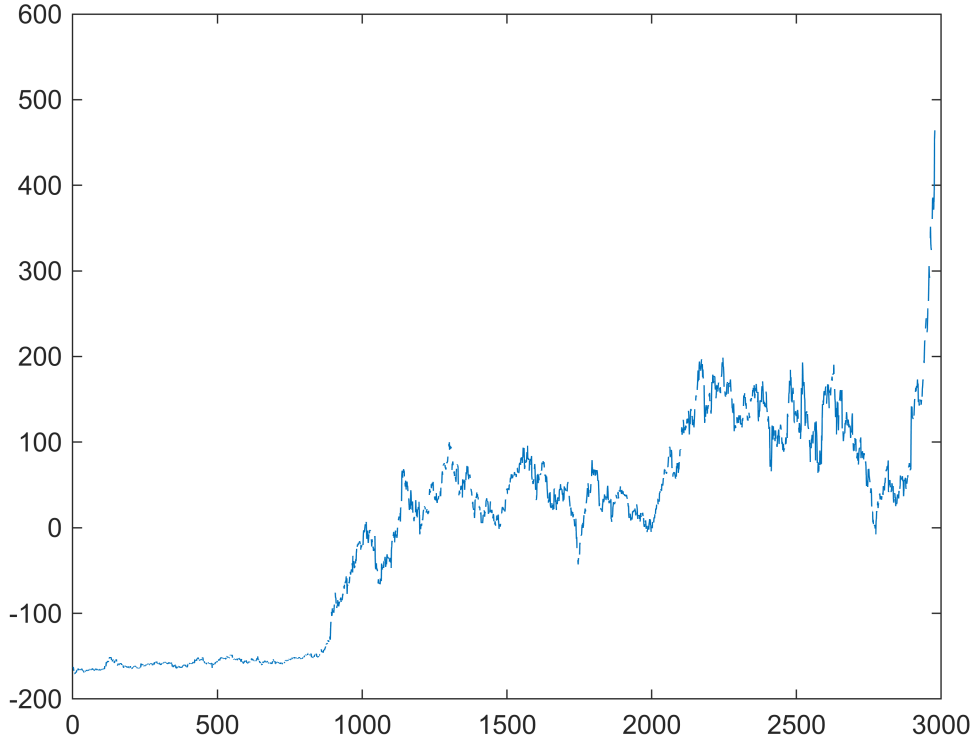
```
Date2=datetime(Date,'InputFormat','dd-MM-yyyy');
Close1=Close-mean(Close); %mean centering Data
```

Accounting for missing data my adding NaN whenever the Close Prices are unavailable

```
c=1;
T(c)=Date2(1); %will contain all the dates including dates which don't have close
prices
T2(c)=Close1(1); %will contain close prices corresponding to each day with NaN for
days without prices
c=2;
for i=2:2416
    n=Date2(i)-Date2(i-1);
    %Date2(i-1)+n-1
    n=n-1;
    j=1;
    while j<n
        T(c)=Date2(i-1)+j;
        T2(c)=NaN;
        c=c+1;
        j=j+1;
    end
    T(c)=Date2(i);
    T2(c)=Close1(i);
    c=c+1;
end
```

Visualizing the data set with gaps for missing data

```
plot(T2')
```



Splitting the data set into Test and Train data sets

```
r = randi([1 2980],400,1);
r=sort(r)
```

```
r = 400×1
     4
     9
    14
    15
    21
    26
    27
    31
    45
    47
     ⋮
```

```
r=unique(r); %removing repeated indices
j=1;
for i=1:2980
    if j<height(r) && r(j)==i
        test(i)=T2(i);
        train(i)=NaN;
        j=j+1;
    else
        test(i)=NaN;
```

```
        train(i)=T2(i);
    end
end
```

Fitting a seasonal ARIMA model as we see that data repeats trends every year

```
Mdl = arima(Constant=0,D=1,Seasonality=12,MALags=1,SMALags=12)
```

```
Mdl = 
  arima with properties:

     Description: "ARIMA(0,1,1) Model Seasonally Integrated with Seasonal MA(12) (Gaussian Distribution)"
      SeriesName: "Y"
    Distribution: Name = "Gaussian"
               P: 13
               D: 1
               Q: 13
        Constant: 0
              AR: {}
             SAR: {}
              MA: {NaN} at lag [1]
             SMA: {NaN} at lag [12]
     Seasonality: 12
            Beta: [1×0]
        Variance: NaN
```

```
EstMdl = estimate(Mdl,train')
```

```
    ARIMA(0,1,1) Model Seasonally Integrated with Seasonal MA(12) (Gaussian Distribution):
```

	Value	StandardError	TStatistic	PValue
Constant	0	0	NaN	NaN
MA{1}	0.031777	0.015113	2.1027	0.035491
SMA{12}	-0.976	0.0053148	-183.64	0
Variance	58.704	0.38497	152.49	0

```
EstMdl = 
  arima with properties:

     Description: "ARIMA(0,1,1) Model Seasonally Integrated with Seasonal MA(12) (Gaussian Distribution)"
      SeriesName: "Y"
    Distribution: Name = "Gaussian"
               P: 13
               D: 1
               Q: 13
        Constant: 0
              AR: {}
             SAR: {}
              MA: {0.0317774} at lag [1]
             SMA: {-0.975999} at lag [12]
     Seasonality: 12
            Beta: [1×0]
        Variance: 58.7042
```

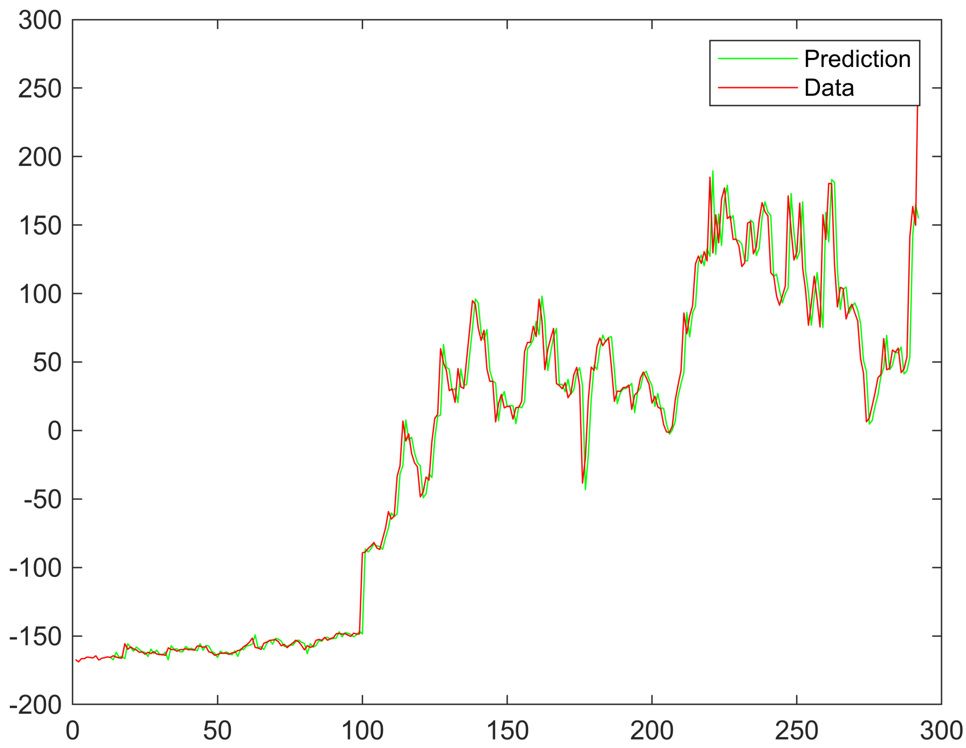
```
Residual = infer(EstMdl,test'); %Residuals obtained after testing model on test
data set
j=1;
```

```
test = test(~isnan(test)); % remove NaNs
test'
```

```
ans = 292×1
 -167.2036
 -168.9437
 -166.5637
 -166.5137
 -165.4037
 -165.6837
 -166.0537
 -164.4536
 -167.6136
 -166.2737
     ⋮
```

Plotting the Test data set vs the prices predicted by ARIMA model

```
Pred=test'- Residual;
plot(Pred,'green')
hold on
plot(test,'red')
legend("Prediction", "Data")
hold off
```



Calculating NMSE for test data set

```
Pred_m=mean(abs(Pred));
Test_m=mean(abs(test));
N=height(test')
```

```
N = 292
```

```
sum_sqaure=sum(Residual.^2);
NMSE=sum_sqaure/(N*Pred_m*Test_m)
```

```
NMSE = 0.0359
```

Calculating MAPE for test data set

```
Diff=sum(abs(abs(Residual)./Pred));
MAPE=Diff*100/N
```

```
MAPE = 27.1257
```

We notice that both the NMSE value and the MAPE value in the case of ARIMA model is much lower than the lombscale periodogram value. This shows that the ARIMA performs much better than the Lombscale Periodogram by taking into account seasonalities too.