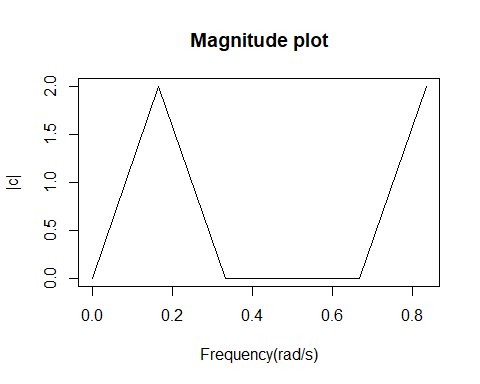
Assignment\_4

Shritej Chavan (BE14B004)

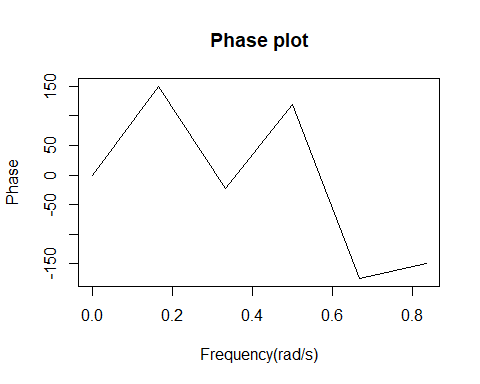
October 8, 2018

### Q.1 Discrete Time Fourier Series

Np = 6 # Fundamental time period  
k = seq(0,Np)  
x = 4\*sin(pi\*(k-2)/3)   
c = rep ( 0 ,Np) #Intialise to zeros  
for (h in 1 :Np) {  
 for (h1 in 1 :Np) {  
 c[h] = c[h] + (x[h1]\*exp(-1i\*2\*pi\*(h1-1)\*(h-1)/Np))  
 }  
 c[h] = c[h]/Np   
}  
plot(seq(0 ,1-(1/Np),1/Np),abs(c),type="l",main = "Magnitude plot", xlab = "Frequency(rad/s)", ylab = "|c|")

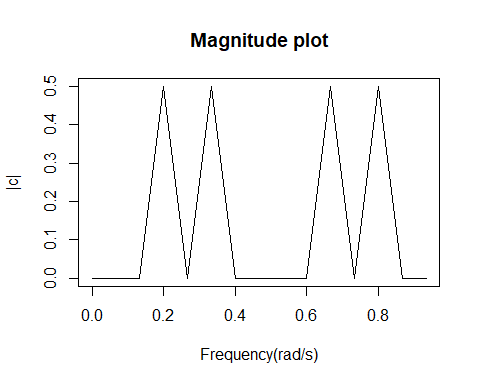


frame()  
plot(seq(0,1-(1/Np), 1/Np), Arg(c)\*180/pi, type = "l", main = "Phase plot", xlab = "Frequency(rad/s)", ylab = "Phase")

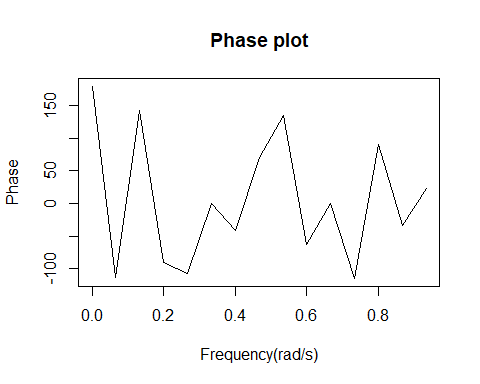


1. The above signal has fundamental period of 15 samples. The fourier coefficients of the signal are given by

Np = 15 # Fundamental time period  
k = seq(0,Np)  
x = cos(2\*pi\*k/3) + sin(2\*pi\*k/5)   
c = rep ( 0 ,Np) #Intialise to zeros  
for (h in 1 :Np) {  
 for (h1 in 1 :Np) {  
 c[h] = c[h] + (x[h1]\*exp(-1i\*2\*pi\*(h1-1)\*(h-1)/Np))  
 }  
 c[h] = c[h]/Np   
}  
plot(seq(0 ,1-(1/Np),1/Np),abs(c),type="l",main = "Magnitude plot", xlab = "Frequency(rad/s)", ylab = "|c|")

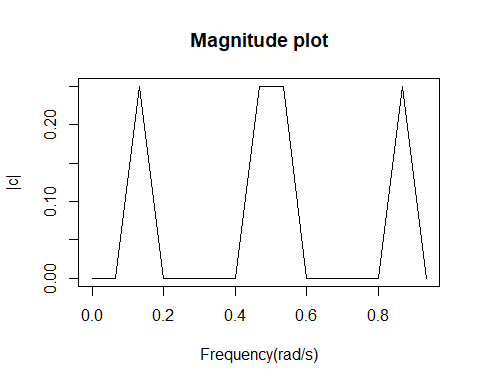


frame()  
plot(seq(0,1-(1/Np), 1/Np), Arg(c)\*180/pi, type = "l", main = "Phase plot", xlab = "Frequency(rad/s)", ylab = "Phase")

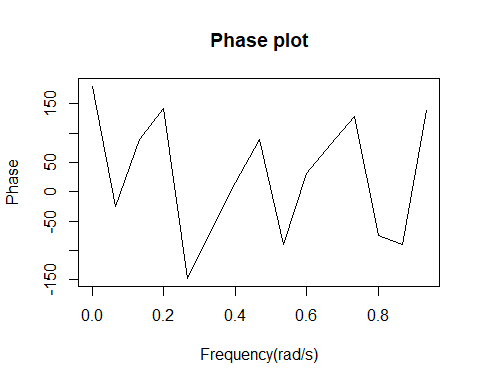


1. Therefore,

Np = 15 # Fundamental time period  
k = seq(0,Np)  
x = 0.5\*(sin(16\*pi\*k/15)- sin(4\*pi\*k/15))  
c = rep ( 0 ,Np) #Intialise to zeros  
for (h in 1 :Np) {  
 for (h1 in 1 :Np) {  
 c[h] = c[h] + (x[h1]\*exp(-1i\*2\*pi\*(h1-1)\*(h-1)/Np))  
 }  
 c[h] = c[h]/Np   
}  
plot(seq(0 ,1-(1/Np),1/Np),abs(c),type="l", main = "Magnitude plot", xlab = "Frequency(rad/s)", ylab = "|c|")



frame()  
plot(seq(0,1-(1/Np), 1/Np), Arg(c)\*180/pi, type = "l", main = "Phase plot", xlab = "Frequency(rad/s)", ylab = "Phase")



The fundamental period Np is 8 samples. The series is given by

Calculating coefficients in R, we get

starting from

1. Periodic Signal $x[k] = {1,0,1,2,3,2}

The Parseval’s identity for a periodic signal is given by

where is Fourier coefficient of . Substituting given series in the equation, we get

and

Hence, Parseval’s identity has been proved.

### Q.2 Fourier Transform and FRF

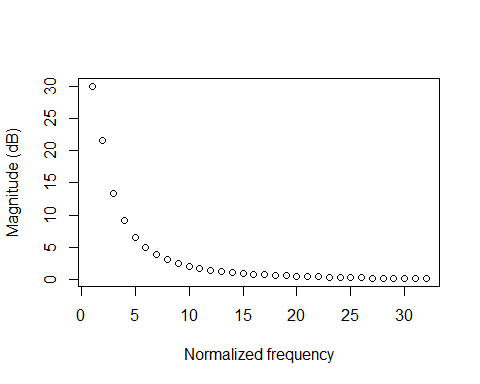
1. X(0)

Since the sequence is even imaginary terms cancel out

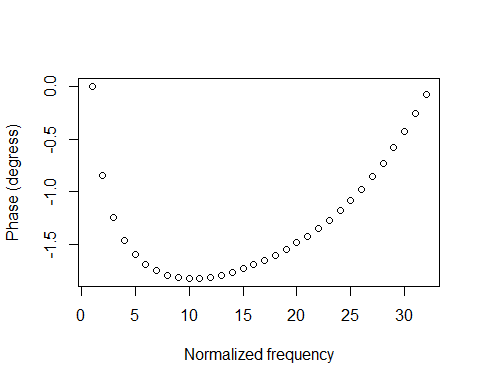
Frequency response function:

c)

frf = function(x)(1 + 0.5\*exp(-1i\*x))/(1 - 1.4\*exp(-1i\*x) + 0.45\*exp(-2i\*x))  
x = seq(from = 0, to = pi, by = 0.1)  
plot(abs(frf(x)), ylab = "Magnitude (dB)", xlab = "Normalized frequency")



plot(Arg(frf(x)), ylab = "Phase (degress)", xlab = "Normalized frequency")

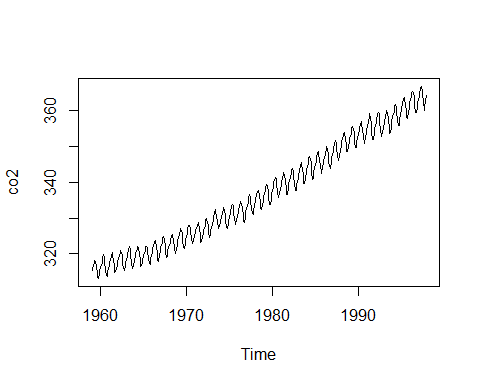


From the above plots, we can conclude that it is a low pass filter.

### Q.3 Modelling non- stationary process

The given co2 data is as follows

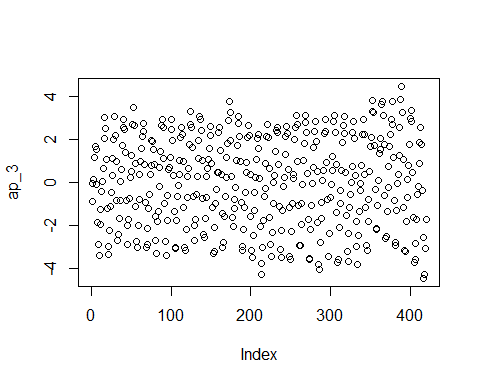
plot(co2)



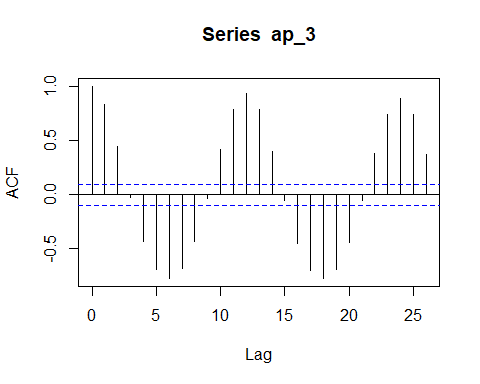
train = co2[1:420]  
  
  
##co2.stl <- stl(train,s.window = "periodic")  
#plot(co2.stl)  
  
lin = 1:length(train)  
ap <- lm(train~poly(lin,3)) #Fitting the polynomial of order 3  
print(ap)

##   
## Call:  
## lm(formula = train ~ poly(lin, 3))  
##   
## Coefficients:  
## (Intercept) poly(lin, 3)1 poly(lin, 3)2 poly(lin, 3)3   
## 334.251 263.306 29.209 -8.685

ap\_3 = ap$residuals  
plot(ap\_3)

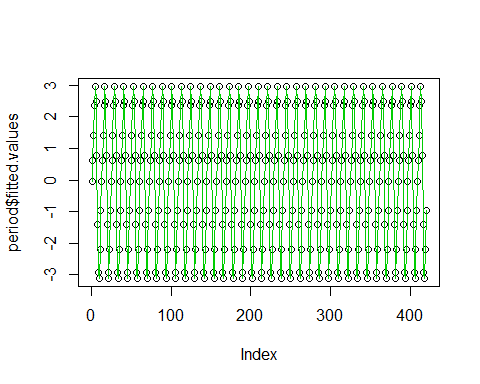


acf(ap\_3)



From the above plot, we can say that the residuals obtained after fitting the data with polynomial of degree 3 has seasonality of 12 which is obvious given the annually repeated pattern.

period = lm(ap\_3 ~ sin(2\*pi\*lin/12)+I(cos(2\*pi\*lin/12))+I(sin(4\*pi\*lin/12))+I(cos(4\*pi\*lin/12)))  
#Including the 2nd and 3rd harmonic   
  
plot(period$fitted.values)  
lines(fitted(period)~lin,col=3)

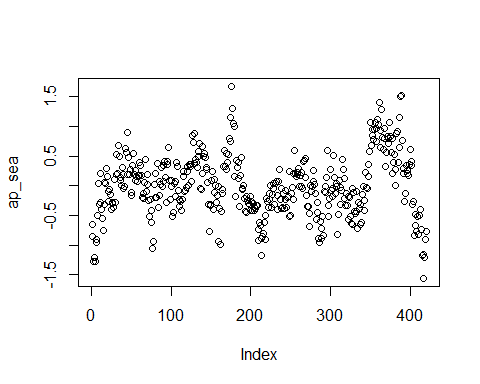


The above plot of sum of sines and cosines of the 1st and 2nd harmonic has been fitted to the Residual data.

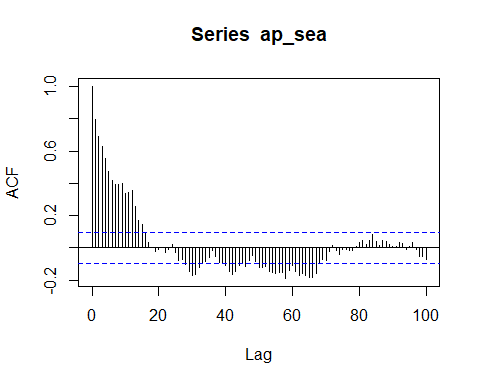
summary(period)

##   
## Call:  
## lm(formula = ap\_3 ~ sin(2 \* pi \* lin/12) + I(cos(2 \* pi \* lin/12)) +   
## I(sin(4 \* pi \* lin/12)) + I(cos(4 \* pi \* lin/12)))  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -1.5599 -0.3244 -0.0171 0.3169 1.6776   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 8.114e-17 2.536e-02 0.000 1.000   
## sin(2 \* pi \* lin/12) 2.159e+00 3.587e-02 60.205 <2e-16 \*\*\*  
## I(cos(2 \* pi \* lin/12)) -1.727e+00 3.587e-02 -48.155 <2e-16 \*\*\*  
## I(sin(4 \* pi \* lin/12)) -8.085e-03 3.587e-02 -0.225 0.822   
## I(cos(4 \* pi \* lin/12)) 7.619e-01 3.587e-02 21.242 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.5198 on 415 degrees of freedom  
## Multiple R-squared: 0.9391, Adjusted R-squared: 0.9385   
## F-statistic: 1599 on 4 and 415 DF, p-value: < 2.2e-16

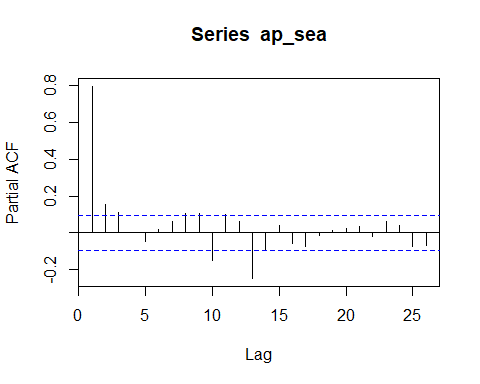
ap\_sea = period$residuals  
plot(ap\_sea)



acf(ap\_sea, lag.max = 100)



pacf(ap\_sea)



Observing the PACF of the residual obtained after fitting the trend and seasonality we can vaguely conclude the process is AR(13)

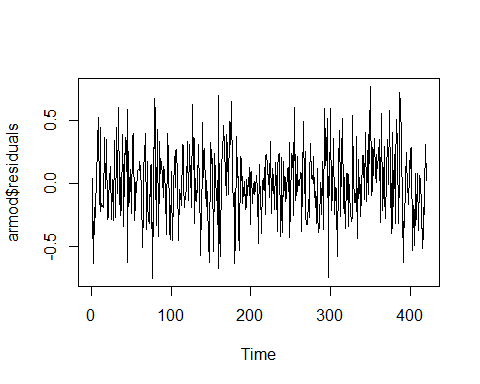
armod = arima(ap\_sea, c(13,0,0))  
armo = ar(ap\_sea, aic = TRUE, order.max = 13)  
print(armod)

##   
## Call:  
## arima(x = ap\_sea, order = c(13, 0, 0))  
##   
## Coefficients:  
## ar1 ar2 ar3 ar4 ar5 ar6 ar7 ar8  
## 0.6739 0.0912 0.0508 0.0739 -0.0653 -0.0222 0.0101 0.0069  
## s.e. 0.0470 0.0564 0.0565 0.0559 0.0555 0.0553 0.0553 0.0554  
## ar9 ar10 ar11 ar12 ar13 intercept  
## 0.1933 -0.1892 0.0809 0.2292 -0.2558 -0.0390  
## s.e. 0.0555 0.0562 0.0572 0.0572 0.0483 0.1127  
##   
## sigma^2 estimated as 0.08032: log likelihood = -67.72, aic = 165.44

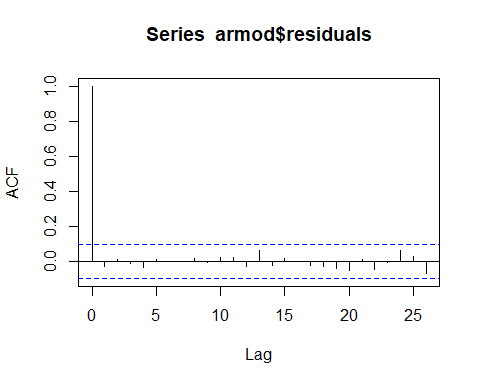
print(armo)

##   
## Call:  
## ar(x = ap\_sea, aic = TRUE, order.max = 13)  
##   
## Coefficients:  
## 1 2 3 4 5 6 7 8   
## 0.6806 0.1012 0.0454 0.0629 -0.0741 -0.0272 0.0013 0.0147   
## 9 10 11 12 13   
## 0.1890 -0.1978 0.0803 0.2278 -0.2474   
##   
## Order selected 13 sigma^2 estimated as 0.08596

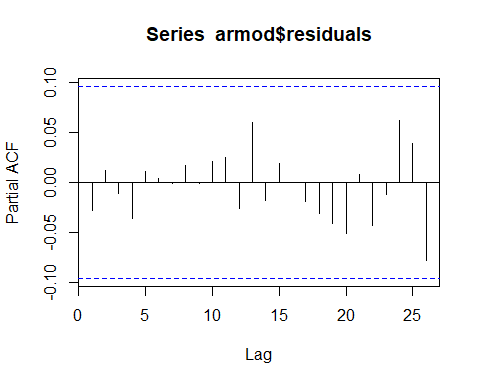
plot(armod$residuals)



acf(armod$residuals)



pacf(armod$residuals)



Above we can see that from both method (‘arima’ and ‘ar’ function) AR(13) model has been chosen comparing the AIC’s of all the AR model upto order 13.

Also, we can clearly see that the AR(13) model overfits the data but the residuals obtained after fitting the AR(13) model satisfies the white noise characteristics.

Hence, comprising the overfitting test we chose AR(13) model to fit the stationary process of the given co2 data.

where,

Now we cross-validate the model on the remaining dataset of size N=48, we get

set.seed(21)  
t = 1:468  
test = co2[421:468]  
#vk = arima.sim(468, model = list(ar = c(-0.6379, -0.0912, -0.0508, -0.0739, 0.6653,0.0222,-0.0101,-0.0069, -0.1933, 0.1892, -0.0809, -0.2292, 0.2558), ma = c(0,0,0)))   
  
#Error : model not stationary  
  
seasonal = 2.159\*sin(2\*pi\*t/12) - 1.727\*cos(2\*pi\*t/12) - 0.008085\*sin(2\*pi\*t/6) + 0.7619\*cos(2\*pi\*t/6)  
  
#xk = 334.251 + 263.306\*t + 29.209\*t^2 - 8.685\*t^3 + vk + seasonal  
  
#plot(xk)

### Q.4 ACVF from PSD

The p.s.d of a stationary process is given by

We know that,

Therefore factorization gives,

We can conclude that,

It is an AR(1) process and

Therefore,

Similarly,

Using the inverse Fourier transform in R

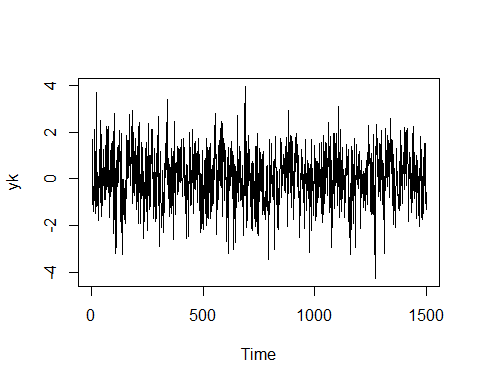
f = seq(from = -0.5, to = 0.5, by = 0.01)  
psd = 1.44/(1.16 - 0.8\*cos(2\*pi\*f))  
invft = abs(fft(psd , inverse = TRUE))/length(psd)  
invft[1:5]

## [1] 1.70458679 0.68359278 0.27712131 0.11133245 0.04523069

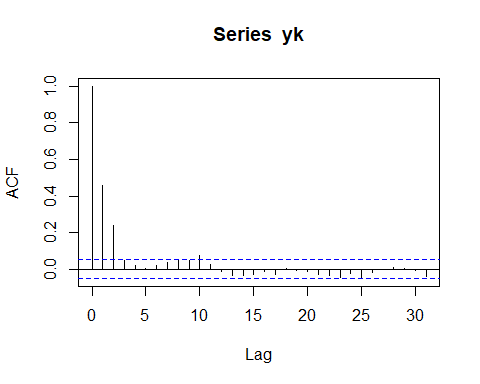
### Q.4 Power spectral density estimation

Simulating series

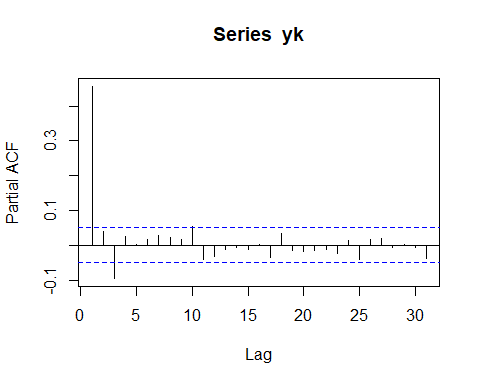
#Simulating series  
  
set.seed(323)  
yk = arima.sim(model = list(ar = c(0,0.25), ma = 0.45), 1500)  
  
plot(yk)



acf(yk)



pacf(yk)

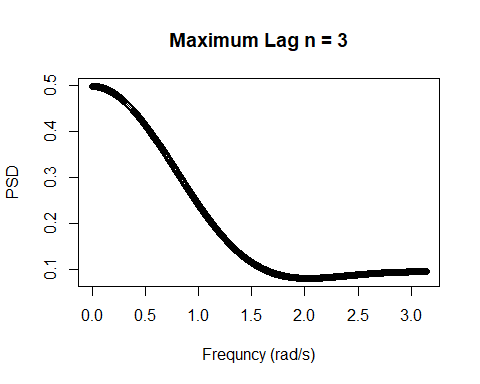


1. Estimating the ACVF and using W-K theorem

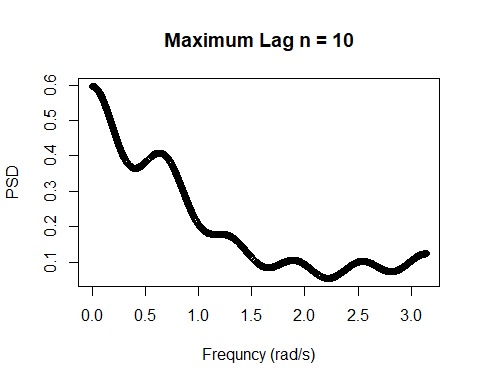
From the Weiner-Khinchin Theorem,

Usually, we do not have estimates of the ACVF at large lags, However, the ACVF of a stationary time series dies down after few lags. Therefore we can truncate the above expression to the first few lags.

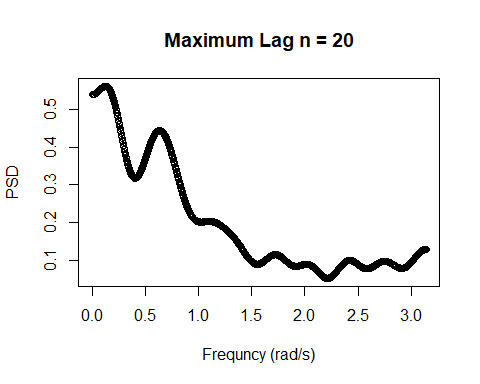
#PSD from Weiner-Khinchin  
  
psd\_acf = function(w, acvf, lag\_max){  
   
 k = 1:lag\_max  
 psd\_ = (acvf[1] + 2\*sum(acvf[-1]\*cos(w\*k)))/(2\*pi)  
 psd\_  
}  
  
w = seq(0,pi,length = 1000)  
lag\_max = c(3,10,20,40)  
wk\_gamma = matrix(nrow = 1000, ncol = 4)  
  
for (i in seq\_along(lag\_max)) {  
 acvf = acf(yk, lag\_max[i], type = "covariance", plot = FALSE)  
 wk\_gamma[,i] = sapply(w, psd\_acf, acvf$acf, lag\_max[i])  
   
   
}  
plot( w, wk\_gamma[,1], main = "Maximum Lag n = 3", xlab = "Frequncy (rad/s)", ylab = "PSD")



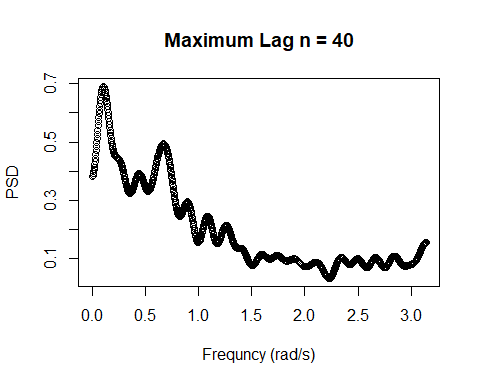
plot( w, wk\_gamma[,2], main = "Maximum Lag n = 10",xlab = "Frequncy (rad/s)", ylab = "PSD")



plot( w, wk\_gamma[,3], main = "Maximum Lag n = 20",xlab = "Frequncy (rad/s)", ylab = "PSD")



plot( w, wk\_gamma[,4], main = "Maximum Lag n = 40",xlab = "Frequncy (rad/s)", ylab = "PSD")



As we can see from the above graph, with the first method PSD is sensitive to the maximum lag and the noise increases as maximum lag is increased.

ii)Estimation from Time-series model

Observing the ACF and PACF plot, we see that the ACF plot dies down after lag 2 and PACF plot dies after lag 3. Therefore we choose to fit an ARMA(3,2) model.

#PSD from time series modelling  
  
arma = arima(yk, order= c(3,0,2))  
print(arma)

##   
## Call:  
## arima(x = yk, order = c(3, 0, 2))  
##   
## Coefficients:  
## ar1 ar2 ar3 ma1 ma2 intercept  
## 0.1648 0.1527 -0.0616 0.2797 0.0501 0.0601  
## s.e. 0.3343 0.1990 0.1351 0.3348 0.2668 0.0457  
##   
## sigma^2 estimated as 0.9835: log likelihood = -2116.1, aic = 4246.19

We can see that the coefficients are insignificant. Therefore this is not the right model. Now we fit an AR(3) model.

armod3 = arima(yk, order = c(3,0,0))  
  
armod1 = arima(yk, order = c(1,0,0))  
print(armod3)

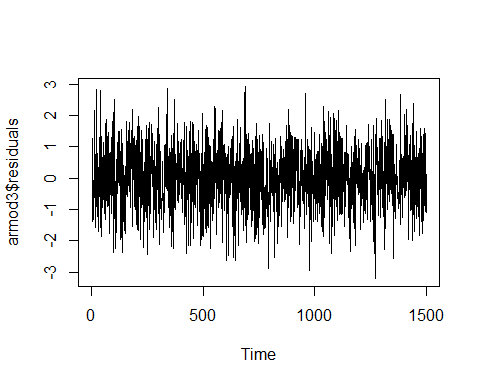
##   
## Call:  
## arima(x = yk, order = c(3, 0, 0))  
##   
## Coefficients:  
## ar1 ar2 ar3 intercept  
## 0.4422 0.0805 -0.0948 0.0601  
## s.e. 0.0257 0.0280 0.0257 0.0448  
##   
## sigma^2 estimated as 0.9842: log likelihood = -2116.57, aic = 4243.15

print(armod1)

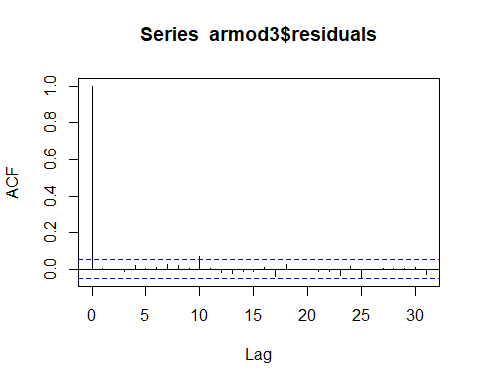
##   
## Call:  
## arima(x = yk, order = c(1, 0, 0))  
##   
## Coefficients:  
## ar1 intercept  
## 0.4563 0.0602  
## s.e. 0.0230 0.0473  
##   
## sigma^2 estimated as 0.9946: log likelihood = -2124.49, aic = 4254.97

Among the above 2 models, AR(3) seems to be a better fit Now checking for underfitting

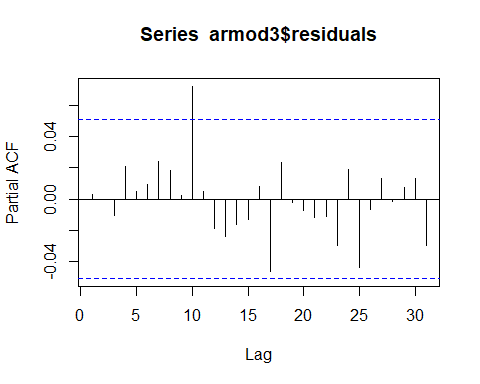
plot(armod3$residuals)



acf(armod3$residuals)

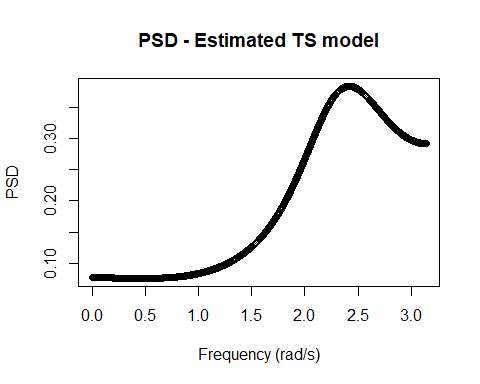


pacf(armod3$residuals)

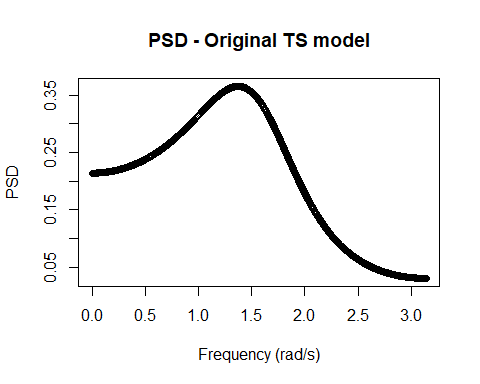


The above figures does indicate WN characteristics, So AR(3) is almost a perfect fit

#transfer function  
  
H = function(w)1/(1 + 0.4422\*exp(-1i\*w) + 0.0805\*exp(-2i\*w) - 0.0948\*exp(-3i\*w))  
  
gamma\_mod = abs(sapply(w,H))^2\*0.9842/(2\*pi)  
  
#actual transfer function  
  
H\_actual = function(w)(1 + 0.45\*exp(-1i\*w))/(1 + 0.25\*exp(-2i\*w))  
  
gamma\_actual = abs(sapply(w,H\_actual))^2/(2\*pi)  
  
plot(w, gamma\_mod, main = "PSD - Estimated TS model", xlab = "Frequency (rad/s)", ylab = "PSD")



plot(w, gamma\_actual,main = "PSD - Original TS model", xlab = "Frequency (rad/s)", ylab = "PSD")



We can clearly see there is a difference between original and estimated model plot of PSD vs frequency

To conclude

The first method being the non-parametric approach is easier to use since we don’t have to fit any model. As we saw earlier the estimate is maximum lag sensitive and hence noisy. In the second approach we have less parameters to fit. However fitted model maybe different from the original data generating process and may result in incorrect estimates.