STAT 720 sp 2019 hw 3

due on Wednesday, March 20th, 2019

Please make use of my R package `tscourse` to complete the homework. You can install the package with the following commands:

```
library(devtools)
devtools::install_github("gregorkb/tscourse")
library(tscourse)
```

This pulls the package from where it resides in my github repository. You will first need to install the `devtools` package using `install.packages("devtools")`.

1. Do problems 5.1, 5.2, and 5.4 of B&D Intro.

5.1 The following R code makes the computations.

```
n <- 100
g0 <- 1382.2
g1 <- 1114.4
g2 <- 591.73
g3 <- 96.216 # not needed

phi1.hat <- (g0*g1 - g1*g2)/(g0^2 - g1^2)
phi2.hat <- (g0*g2 - g1*g1)/(g0^2 - g1^2)
sigma2.hat <- g0 - (phi1.hat * g1 + phi2.hat * g2)

phi1.lo <- phi1.hat - 1.96 * sqrt(sigma2.hat/n) * sqrt( g0 / ( g0^2 - g1^2) )
phi1.up <- phi1.hat + 1.96 * sqrt(sigma2.hat/n) * sqrt( g0 / ( g0^2 - g1^2) )

phi2.lo <- phi2.hat - 1.96 * sqrt(sigma2.hat/n) * sqrt( g0 / ( g0^2 - g1^2) )
phi2.up <- phi2.hat + 1.96 * sqrt(sigma2.hat/n) * sqrt( g0 / ( g0^2 - g1^2) )
```

We get

```
> phi1.hat
[1] 1.31755
> phi2.hat
[1] -0.6341682
> sigma2.hat
[1] 289.1791
>
> phi1.lo
```

```
[1] 1.166003
> phi1.up
[1] 1.469096
>
> phi2.lo
[1] -0.7857144
> phi2.up
[1] -0.4826219
```

5.2

2. Let $\{X_t, t \in \mathbb{Z}\}$ be the ARMA$(1,1)$ times series defined by

$$X_t - \phi X_{t-1} = Z_t + \theta Z_{t-1},$$

where $\{Z_t, t \in \mathbb{Z}\}$ is WN$(0, \sigma^2)$ and $\theta + \phi \neq 0$ and $|\phi| \neq 1$.

(a) Give an expression for the spectral density of the time series in terms of ϕ, θ, and σ^2.

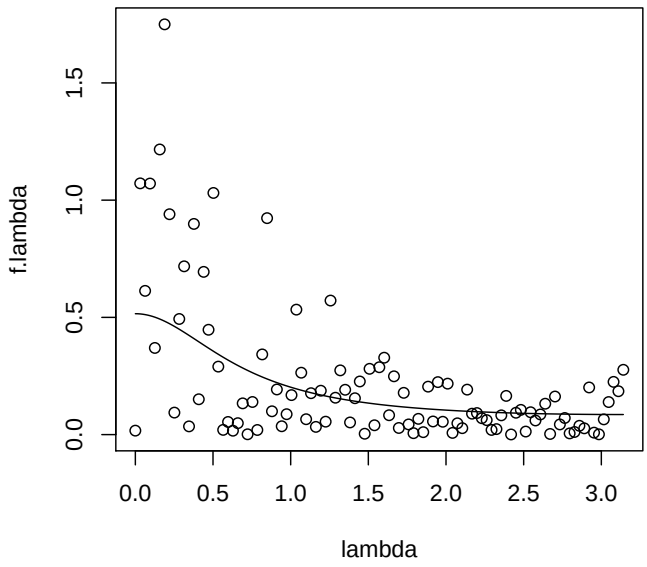
The spectral density is given by

$$f(\lambda) = \frac{\sigma^2}{2\pi} \frac{1 + 2\theta\cos(\lambda) + \theta^2}{1 - 2\phi\cos(\lambda) + \phi^2}, \quad -\pi < \lambda \leq \pi.$$

(b) Choose some values of ϕ, θ, and σ^2 and generate a realization from the model of length $n = 200$. Then compute from the data the periodogram ordinates at the Fourier frequencies

$$\frac{k}{n} \cdot 2\pi, \quad k = -\left\lfloor \frac{n-1}{2} \right\rfloor, \ldots, \left\lfloor \frac{n}{2} \right\rfloor$$

and plot the ordinates, scaled by $1/(2\pi)$, against the frequencies with the true spectral density function overlaid. Note: Since the periodogram and the spectral density are even functions, it is only necessary to consider their values over the range $[0, \pi]$. My plot looks like the following (the spectral density will change depending on what values you choose for ϕ, θ, and σ^2):

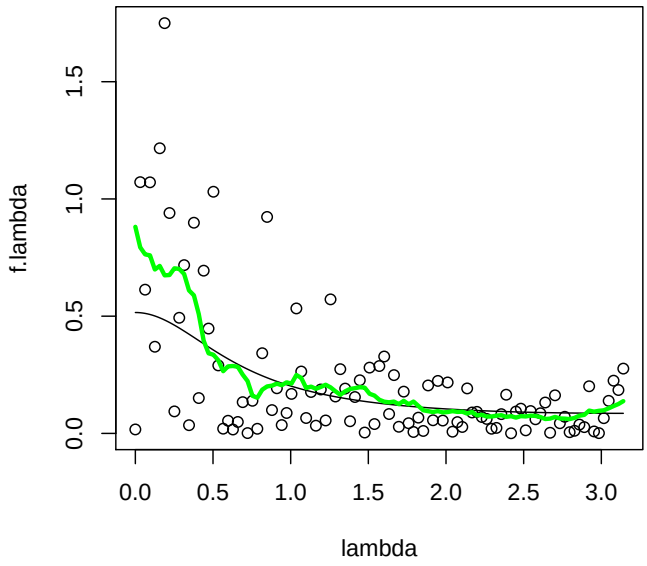


The following R code does this:

```
# choose values for phi, theta, and sigma
phi <- .5
theta <- -.1
sigma <- 1

# evaluate true spectral density function
lambda <- seq(0,pi,length=200)
f.lambda <- sigma^2/(2*pi) * ( 1 + 2 * theta * cos(lambda) + theta^2 )
            / ( 1 - 2 * phi * cos(lambda) + phi^2)

# generate data
n <- 200
X <- get.ARMA.data(phi,theta,sigma,n)

# define Fourier frequencies
lambda.F <- (-floor((n-1)/2):floor(n/2))/n*2*pi

# get DFT and periodogram
E <- matrix(NA,n,n)
for(i in 1:n)
{
        E[i,] <- 1/sqrt(n) * exp(1i*i*lambda.F)
}
D <- as.vector(t(Conj(E)) %*% X)
```

```
I <- Mod(D[lambda.F>=0])^2

# make plot
plot(f.lambda~lambda,type="l",ylim=range(I/(2*pi)))
points(I/(2*pi)~lambda.F[lambda.F>=0])
```

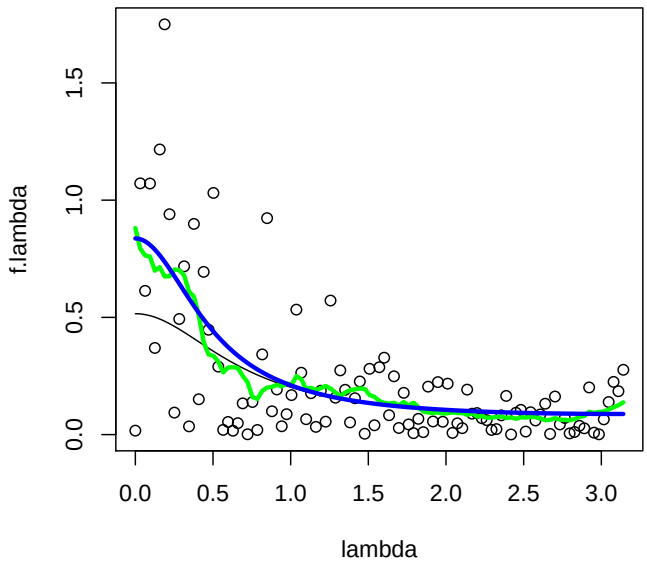
(c) Try smoothing the periodogram ordinates using the function `ksmooth()` under default settings. This takes local averages, much like the way we previously estimated trends. Add to the plot from part (b) the curve resulting from locally averaging the periodogram. My plot looks like the following:



We can plot the smoothed periodogram with the following:

```
plot(f.lambda~lambda,type="l",ylim=range(I/(2*pi)))
points(I/(2*pi)~lambda.F[lambda.F>=0])
lines(ksmooth(x=lambda.F[lambda.F>=0],y=I/(2*pi)),col="green",lwd=3)
```

(d) Compute the maximum likelihood estimators of ϕ and θ based on the data. Then plug these estimates into the expression from part (a) to obtain a maximum-likelihood-based estimator of the spectral density. Add to the plot from part (b) the curve corresponding to this estimator of the spectral density. My plot looks like the following:




```
# get mles from arima() function
arima.out <- arima(x = X - mean(X),order=c(1,0,1),include.mean=FALSE)
phi.hat <- arima.out$coef[1]
theta.hat <- arima.out$coef[2]
sigma.hat <- arima.out$sigma2

# get mle-based estimator of the spectral density
f.lambda.mle <- sigma.hat^2/(2*pi)*(1 + 2*theta.hat*cos(lambda) + theta.hat^2)
                / ( 1 - 2 * phi.hat * cos(lambda) + phi.hat^2)

# make plot
plot(f.lambda~lambda,type="l",ylim=range(I/(2*pi)))
points(I/(2*pi)~lambda.F[lambda.F>=0])
lines(ksmooth(x=lambda.F[lambda.F>=0], y=I/(2*pi)),col="green",lwd=3)
lines(f.lambda.mle~lambda,col="blue",lwd=3)
```

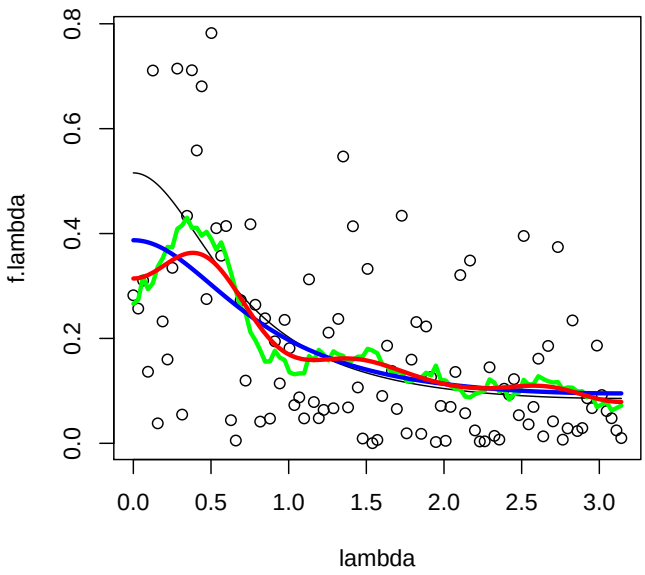
(e) Now compute a lag-window estimator of the spectral density. In particular, compute the estimator

$$\hat{f}(\lambda) = \frac{1}{2\pi} \sum_{h=-L}^{L} w(h/L)\hat{\gamma}_n(h)e^{-\imath h\lambda}, \quad -\pi < \lambda \leq \pi,$$

using the lag-window function given by

$$w(x) = \begin{cases} 1 - 6x^2 + 6|x|^3, & |x| < 1/2 \\ 2(1 - |x|)^3, & 1/2 \leq |x| \leq 1 \\ 0, & |x| > 1, \end{cases}$$

which is the Parzen window. Compute this estimator using $L = \lfloor \sqrt{n} \rfloor$, and add to the plot from part (b) the resulting curve. *I you wish, you may submit one single plot for parts* (b)–(e). My plot looks like the following:



The following R code computes the lag-window estimator:

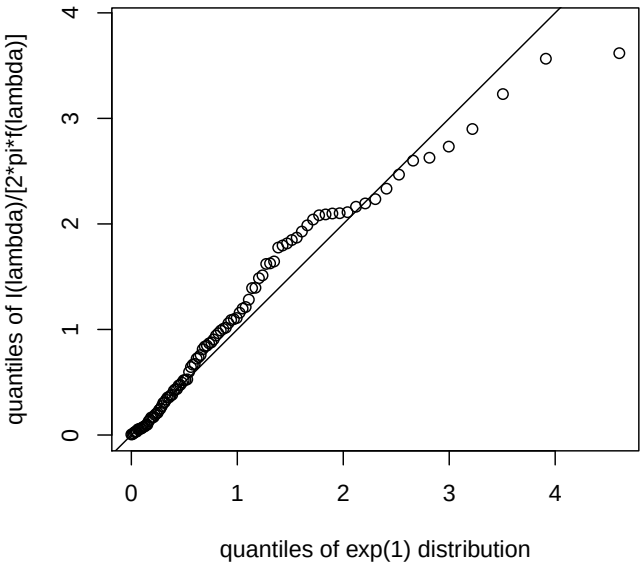
```
# define the parzen window
parzen <- function(x)
{
        if( abs(x) < 1/2){

                return( 1 - 6 * x^2 + 6 * abs(x)^3 )

        } else if( (abs(x) >= 1/2) & (abs(x) <= 1)){

                return( 2*(1 - abs(x))^3 )

        } else {

                return( 0 )

        }
}

# define L and get vector of sample autocovariances
L <- floor(sqrt(n))
```

```
gamma.hat.0toL <- sample.acf(X,max.lag = L)$gamma.hat
gamma.hat <- c(gamma.hat.0toL[(L+1):2],gamma.hat.0toL)

# compute lag-window estimator
E.lambda <- matrix(NA,2*L+1,length(lambda))
gamma.hat.weighted <- numeric(2*L+1)
for(j in (-L):L)
{
        E.lambda[j+L+1,] <- exp( -1i * j * lambda )
        gamma.hat.weighted[j+L+1] <- parzen(abs(j/L)) * gamma.hat[j+L+1]
}
f.hat.L <- as.numeric(Re(t(Conj(E.lambda)) %*% gamma.hat.weighted / ( 2*pi )))

# make plot
plot(f.lambda~lambda,type="l",ylim=range(I/(2*pi)))
points(I/(2*pi)~lambda.F[lambda.F>=0])
lines(ksmooth(x=lambda.F[lambda.F>=0], y=I/(2*pi)),col="green",lwd=3)
lines(f.hat.L~lambda,col="red",lwd=3)
```

(f) According to Theorem 10.3.2 of B&D Theory, under some conditions, the periodogram ordinates at any finite number of frequencies $\lambda_1, \ldots, \lambda_m$ should asymptotically behave as independent exponentially distributed random variables with means $2\pi f(\lambda_1), \ldots, 2\pi f(\lambda_m)$, respectively, as $n \to \infty$. For the dataset generated in part (b), standardize the periodogram ordinates at the Fourier frequencies according to

$$I_n(\lambda_j)/(2\pi f(\lambda_j)), \quad j = 1, \ldots, n,$$

where $\lambda_1, \ldots, \lambda_n$ are the Fourier frequencies. Then make a quantile-quantile plot of these values to see if they follow the exponential distribution with mean 1. Note that you only need to consider the frequencies in the interval $[0, \pi]$. My plot looks like this:



The following R code generates the quantile-quantile plot.

```
# get true spectral density at Fourier frequencies
f.lambda.F <- sigma^2/(2*pi) * ( 1 + 2 * theta * cos(lambda.F) + theta^2 )
                / ( 1 - 2 * phi * cos(lambda.F) + phi^2)

# standardize periodogram ordinates to have, asymptotically, the exp(1) dist.
I1 <- I/(2*pi*f.lambda.F[lambda.F>=0])

# make quantile-quantile plot
u <- seq(0,1,length=floor(n/2)+1)
plot(quantile(I1,u)~qexp(u),
    ylab="quantiles of I(lambda)/[2*pi*f(lambda)]",
    xlab="quantiles of exp(1) distribution")
abline(0,1)
```

(g) Since

$$f(0) = \frac{1}{2\pi} \sum_{h=-\infty}^{\infty} \gamma(h) e^{-\imath h(0)} = \frac{1}{2\pi} \sum_{h=-\infty}^{\infty} \gamma(h),$$

we have

$$\sqrt{n}(\bar{X} - \mu) \to \text{Normal}(0, 2\pi f(0)) \text{ in distribution}$$

as $n \to \infty$, where $\bar{X} = n^{-1}(X_1, \ldots, X_n)$ and $\mu = \mathbb{E}\bar{X}$. Run the following simulation: For $n = 200$, generate 500 realizations of length n from a causal invertible ARMA$(1, 1)$ process

with some mean $\mu \neq 0$. For each of the 500 datasets, compute the lag-window estimator

$$\hat{f}(0) = \frac{1}{2\pi} \sum_{h=-L}^{L} w(h/L)\hat{\gamma}_n(h)$$

of $f(0)$, with $w(\cdot)$ and L chosen as in part (e), and use it to construct a 95% confidence interval for μ. Report the coverage of your interval over the 500 simulated datasets and provide your code.

The following R code runs the simulation.

```
S <- 500
mu <- 5
phi <- .5
theta <- -.1
sigma <- 1
n <- 200
L <- floor(sqrt(n))

f.hat.0 <- numeric(S)
X.bar <- lo.ci <- up.ci <- numeric(S)
for( s in 1:S)
{

        X <- get.ARMA.data(phi,theta,sigma,n) + mu
        gamma.hat.0toL <- sample.acf(X,max.lag = L)$gamma.hat
        gamma.hat <- c(gamma.hat.0toL[(L+1):2],gamma.hat.0toL)

        f.hat.0[s] <- sum( sapply(abs((-L):L)/L,FUN=parzen)*gamma.hat)/(2*pi)

        X.bar[s] <- mean(X)
        lo.ci[s] <- X.bar[s] - 1.96 * sqrt( f.hat.0[s] * 2 * pi / n)
        up.ci[s] <- X.bar[s] + 1.96 * sqrt( f.hat.0[s] * 2 * pi / n)

}

mean( (lo.ci < mu) & (up.ci > mu) )
```

My coverage was 0.934.

3. Let $\{X_t, t \in \mathbb{Z}\}$ be the causal AR(p) time series given by

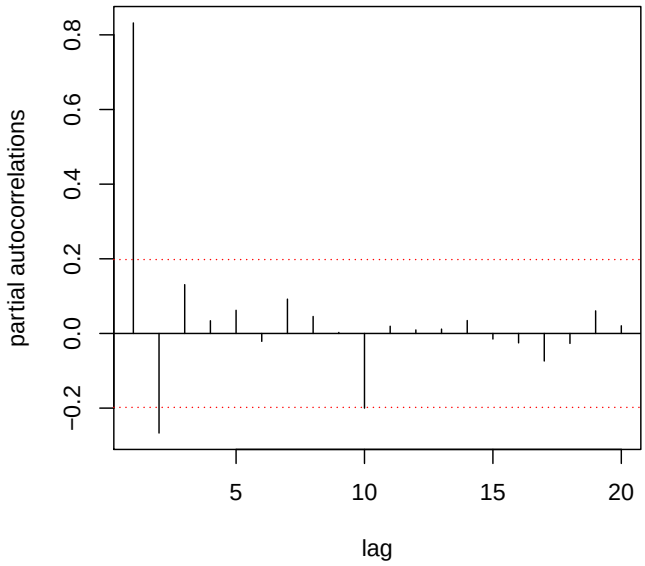
$$X_t - \phi_1 X_{t-1} - \cdots - \phi_p X_{t-p} = Z_t,$$

where $\{Z_t, t \in \mathbb{Z}\}$ is WN$(0, \sigma^2)$, and consider the sequence of Yule-Walker estimators

$$\hat{\boldsymbol{\phi}}_k = (\hat{\phi}_{k,1}, \ldots, \hat{\phi}_{k,k})^T = \hat{\boldsymbol{\Gamma}}_k^{-1} \hat{\boldsymbol{\gamma}}_k,$$

where $\hat{\mathbf{\Gamma}}_k = (\hat{\gamma}_n(i-j))_{1\le i,j\le k}$ and $\hat{\boldsymbol{\gamma}}_k = (\hat{\gamma}_n(1),\ldots,\hat{\gamma}_n(k))^T$ for $k = 1,\ldots,n$, where $\hat{\gamma}_n(\cdot)$ is the sample autocovariance function from a realization $X_1,\ldots,X_n$ of the time series.

(a) The partial autocorrelation function (pacf) at lag k is the value $\hat{\phi}_{k,k}$ for $k = 1,2,\ldots$ We often examine a plot of the pacf to determine the appropriate order of an autoregressive model. Make a plot of the pacf at lags $k = 1,\ldots,20$ based on the Lake Huron data from the R data set `LakeHuron` and add to the plot two horizontal lines at the heights $-n^{-1/2}1.96$ and $n^{-1/2}1.96$. Note that you can extract the values $\hat{\phi}_{1,1},\ldots,\hat{\phi}_{20,20}$ from the output of the Durbin-Levinson algorithm when it is run using the sample autocovariance function. My plot looks like this:



The following R code produces the plot.

```
X <- as.numeric(LakeHuron)
n <- length(X)

gamma.hat <- sample.acf(X,max.lag=n)$gamma.hat
DL.1step.out <- DL.1step(X-mean(X),gamma.hat[1],gamma.hat[-1])

max.p <- 20
pacf.hat <- DL.1step.out$Phi[2:(max.p+1),1]

plot(pacf.hat,type="h", xlab="lag",ylab="partial autocorrelations")
abline(h = 1.96/sqrt(n),lty=3,col="red")
abline(h=0)
abline(h = -1.96/sqrt(n),lty=3,col="red")
```

(b) A convention is to find the smallest lag at which the partial autocorrelations fall outside of the interval $(-n^{-1/2}1.96, n^{-1/2}1.96)$ and to fit an autoregressive model of this order to the data. The purpose of this question is for us to understand why this makes sense.

From Theorem 8.1.2 of B&D Theory, we have, for any $k > p$,

$$\sqrt{n}(\hat{\boldsymbol{\phi}}_k - \boldsymbol{\phi}_k) \to \text{Normal}(0, \boldsymbol{\Gamma}_k^{-1}\sigma^2) \quad \text{in distribution}$$

as $n \to \infty$, where $\boldsymbol{\Gamma}_k = (\gamma(i-j))_{1\le i,j\le k}$ and $\boldsymbol{\phi}_k = (\phi_1, \ldots, \phi_p, \mathbf{0}_{k-p}^T)^T$, where $\mathbf{0}_{k-p}$ is the $(k-p)\times 1$ vector with all entries equal to 0, and, in particular,

$$\sqrt{n}\hat{\phi}_{k,k} \to N(0,1) \quad \text{in distribution}$$

as $n \to \infty$.

i. Show that for any $k > p$, entry (k,k) of the matrix $\boldsymbol{\Gamma}_k^{-1}$ is equal to $(\gamma(0) - \boldsymbol{\phi}_p^T\boldsymbol{\gamma}_p)^{-1}$.
Hint: Consider inverting the matrix

$$\begin{bmatrix} \boldsymbol{\Gamma}_{k-1} & \tilde{\boldsymbol{\gamma}}_{k-1} \\ \tilde{\boldsymbol{\gamma}}_{k-1}^T & \gamma(0) \end{bmatrix},$$

where $\tilde{\boldsymbol{\gamma}}_{k-1} = (\gamma(k-1), \ldots, \gamma(1))^T$, *with a block-matrix inversion formula.*

Using a block inversion formula, we find that entry (k,k) of the matrix $\boldsymbol{\Gamma}_k^{-1}$ is given by

$$\begin{aligned}(\gamma(0) - \tilde{\boldsymbol{\gamma}}_{k-1}^T\boldsymbol{\Gamma}_{k-1}^{-1}\tilde{\boldsymbol{\gamma}}_{k-1})^{-1} &= (\gamma(0) - \tilde{\boldsymbol{\gamma}}_{k-1}^T\tilde{\boldsymbol{\phi}}_{k-1})^{-1} \\ &= (\gamma(0) - \boldsymbol{\gamma}_{k-1}^T\boldsymbol{\phi}_{k-1})^{-1} \\ &= (\gamma(0) - \boldsymbol{\gamma}_p^T\boldsymbol{\phi}_p)^{-1},\end{aligned}$$

where we have used $\tilde{\boldsymbol{\phi}}_{k-1}$ to represent the vector $\boldsymbol{\phi}_{k-1}$ with entries in reverse order; the first equality comes from the Yule-Walker equations and last equality comes from the fact that the last $k-1-p$ entries of $\boldsymbol{\phi}_{k-1}$ are equal to zero.

ii. Show that $\gamma(0) - \boldsymbol{\phi}_p^T\boldsymbol{\gamma}_p = \sigma^2$, so that entry (k,k) of the matrix $\boldsymbol{\Gamma}_k^{-1}\sigma^2$ is equal to 1 for any $k > p$.

We will take the expectation $\mathbb{E}(X_{p+1} - \boldsymbol{\phi}_p^T\tilde{\mathbf{X}}_p)^2$ in two different ways, where $\tilde{\mathbf{X}}_p = (X_p, \ldots, X_1)$. Firstly we have

$$\mathbb{E}(X_{p+1} - \boldsymbol{\phi}_p^T\tilde{\mathbf{X}}_p)^2 = \mathbb{E}Z_{p+1}^2 = \sigma^2.$$

Secondly

$$\begin{aligned}\mathbb{E}\left(X_{p+1} - \boldsymbol{\phi}_p^T\tilde{\mathbf{X}}_p\right)^2 &= \mathbb{E}\left[X_{p+1}^2 - 2\boldsymbol{\phi}_p^T\tilde{\mathbf{X}}_pX_{p+1} + \boldsymbol{\phi}_p^T\tilde{\mathbf{X}}_p\tilde{\mathbf{X}}_p^T\boldsymbol{\phi}_p\right] \\ &= \gamma(0) - 2\boldsymbol{\phi}_p^T\boldsymbol{\gamma}_p + \boldsymbol{\phi}_p^T\boldsymbol{\Gamma}_p\boldsymbol{\phi}_p \\ &= \gamma(0) - 2\boldsymbol{\phi}_p^T\boldsymbol{\gamma}_p + \boldsymbol{\phi}_p^T\boldsymbol{\gamma}_p \quad \text{(Yule-Walker equations)} \\ &= \gamma(0) - \boldsymbol{\phi}_p^T\boldsymbol{\phi}_p.\end{aligned}$$

So entry (k,k) of the matrix $\mathbf{\Gamma}_k^{-1}\sigma^2$ is $(\gamma(0) - \boldsymbol{\phi}_p^T\boldsymbol{\phi}_p)^{-1}\sigma^2 = 1$.

(c) Now explain why it is reasonable to follow the convention described in part (b) for choosing the order of autoregressive model to fit to the data.

For each $k > p$, where p is the true order of the autoregressive model, the partial autocorrelation at lag k is asymptotically Normally distributed with mean 0 and variance 1. If we consider testing the hypothesis H_0: $\phi_k = 0$ versus H_1: $\phi_k \neq 0$, a test which has size asymptotically equal to 0.05 is to reject H_0 if $\hat{\phi}_{k,k}$ falls beyond the upper and lower 0.025 quantile of the standard Normal distribution. So it may be reasonable to choose the order of the autoregressive model as the smallest k for which we reject H_0: $\phi_k = 0$ (Please note, however, that this strategy ignores the problem of multiple testing).

(d) Make a choice for the order of autoregressive model appropriate for the Lake Huron data and compute the Yule-Walker estimators of the autoregressive parameters.

The choice $p = 2$ appears to be appropriate based on the pacf plot. We can extract the Yule-Walker estimates $\hat{\phi}_1$ and $\hat{\phi}_2$ from the output of the Durbin-Levinson algorithm with

```
DL.1step.out$Phi[3,2:1]
```

We get $\hat{\phi}_1 = 1.0538249$ and $\hat{\phi}_2 = -0.2667516$.

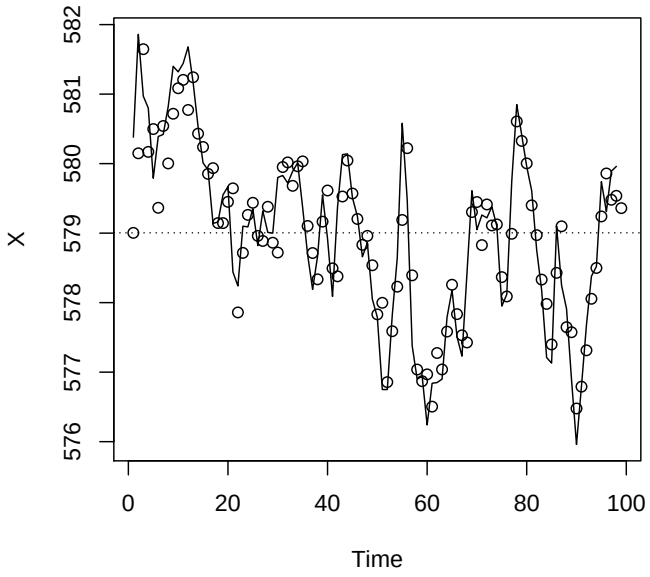
(e) Compute the maximum likelihood estimators of ϕ_1 and ϕ_2 and compare these values to the Yule-Walker estimates.

The R code

```
arima(X - mean(X),order=c(2,0,0),include.mean=FALSE)
```

returns the estimates $\hat{\phi}_1 = 1.0441$ and $\hat{\phi}_2 = -0.2503$. These are quite close to the Yule-Walker estimates.

(f) Use your fitted model to obtain one-step-ahead predictions for $X_1, \ldots, X_n, X_{n+1}$ for the the Lake Huron time series. Make a plot of the original time series with the one-step-ahead predictions overlaid. My plot looks like this:



We can obtain the predictions from the Durbin-Levinson algorithm and make the plot with the following code:

```
X.pred <- DL.1step.out$X.pred + mean(X)
pdf(height=5,width=5,file="hw03_plot06")
plot(X,xlim=c(1,n+1),xlab="Time",type="l",ylim=range(X,X.pred))
points(X.pred)
abline(h=mean(X),lty=3)
```