TMA4285 Time Series Models, Project 1

Kristin Benedicte Bakka and Gunhild Elisabeth Berget

August 30, 2018

In this project the ARMA(1,2) model

$$(1-\phi_1\mathrm{B})Z_t = (1-\theta_1\mathrm{B}-\theta_2\mathrm{B}^2)a_t$$

is considered. $\{a_t\}_{t=-\infty}^{\infty}$ is Gaussian distributed white noise with zero mean and $Var(a_t) = \sigma_a^2$. It is also assumed that $\phi_1 = -0.8$, $\theta_1 = 1.88$, $\theta_2 = -0.98$ and $\sigma_a^2 = 1$.

Writing $\phi(\mathrm{B}) = (1-\phi_1\mathrm{B})$ and $\theta(\mathrm{B}) = (1-\theta_1\mathrm{B}-\theta_2\mathrm{B}^2)$, we have

$$\theta(\mathrm{B}) = 1 - 1.88B + 0.98B^2 = (1-0.99\mathrm{e}^{-0.319i}\mathrm{B})(1-0.99\mathrm{e}^{0.319i}\mathrm{B})\ . \tag{1}$$

The model is stationary and can be expressed in pure MA(∞) form as the root of $\phi(\mathrm{B})$ is $B = -1.25$ is of size greater than 1. The roots of $\theta(B)$ are $1.0101\exp(\pm 0.319i)$. As they lie outside the unit circle the model is invertible and can be expressed in pure $AR(\infty)$ form.

Problem 1

First we find analytical formulas for the variance $Var(Z_t)$ and the autocorrelation function ρ_k. To derive the general autocovariance we write the ARMA$(1,2)$ model as

$$Z_t = \phi_1 Z_{t-1} + a_t - \theta_1 a_{t-1} - \theta_2 a_{t-2}\ ,$$

and multiply both sides with Z_{t-k} and obtain the equation

$$Z_t Z_{t-k} = \phi_1 Z_{t-1}Z_{t-k} + a_t Z_{t-k} - \theta_1 a_{t-1}Z_{t-k} - \theta_2 a_{t-2}Z_{t-k}.$$

The autocovariance of lag k, γ_k is then given by the expected value

$$\gamma_k = \phi_1\gamma_{k-1} + E[a_t Z_{t-k}] - \theta_1 E[a_{t-1}Z_{t-k}] - \theta_2 E[a_{t-2}Z_{t-k}]\ .$$

The variance $\mathrm{Var}(Z_t)$ is the autocovariance of lag $k = 0$

$$\begin{aligned}\gamma_0 &= \phi_1\gamma_1 + E[a_t Z_t] - \theta_1 E[a_{t-1}Z_t] - \theta_2 E[a_{t-2}Z_t] \\ &= \phi_1\gamma_1 + \sigma_a^2 - \theta_1(\phi_1-\theta_1)\sigma_a^2 - \theta_2(-\phi_1\theta_1 - \theta_2 + \phi_1^2)\sigma_a^2 \\ &= \phi_1\gamma_1 + (1 - \phi_1\theta_1 + \theta_1^2 + \phi_1\theta_1\theta_2 + \theta_2^2 - \phi_1^2\theta_2)\sigma_a^2\end{aligned} \tag{2}$$

and to determine it we need the equation for γ_1

$$\begin{aligned}\gamma_1 &= \phi_1\gamma_0 - \theta_1 E[a_{t-1}Z_{t-1}] - \theta_2 E[a_{t-2}Z_{t-1}] \\ &= \phi_1\gamma_0 - \theta_1\sigma_a^2 - \theta_2(\phi_1-\theta_1)\sigma_a^2 \\ &= \phi_1\gamma_0 + (-\theta_1 - \phi_1\theta_2 + \theta_1\theta_2)\sigma_a^2\ .\end{aligned} \tag{3}$$

By inserting (3) in (2) we get the following expression for the variance

$$\mathrm{Var}(Z_t) = \frac{\sigma_a^2}{1-\phi_1^2}[1 - 2\phi_1\theta_1 - 2\phi_1\theta_2(\phi_1 - \theta_1) + \theta_1^2 + \theta_2^2] \,. \tag{4}$$

For $k = 2$

$$\begin{aligned}\gamma_2 &= \phi_1\gamma_1 - \theta_2 E[a_{t-2}Z_{t-2}] \\ &= \phi_1\gamma_1 - \theta_2\sigma_a^2 \,.\end{aligned}$$

Because $E[Z_{t-k}a_{t-i}] = 0$ for $k > i$ the autocovariance for $k > 2$ is given by

$$\gamma_k = \phi_1\gamma_{k-1}$$

Then using the values for ϕ_1, θ_1, θ_2 and σ_a we obtain

$$\gamma_k = \begin{cases} 35.2918 \,, & k = 0 \\ -32.7398 \,, & k = 1 \\ 27.1719 \,, & k = 2 \\ -0.8\gamma_{k-1} \,, & k > 2 \end{cases}$$

Noting that $\rho_k = \frac{\gamma_k}{\gamma_0}$ the values of ρ_k for our model are

$$\rho_k = \begin{cases} 1 \,, & k = 0 \\ -0.9277 \,, & k = 1 \\ 0.7699 \,, & k = 2 \\ -0.8\rho_{k-1} \,, & k > 2 \end{cases} \tag{5}$$

The partial acf of a Gaussian process is the correlation between Z_t and Z_{t+k} conditioned on the intermediate values, that is

$$P_k = \mathrm{corr}(\mathrm{Z}_t, \mathrm{Z}_{t+k} \,|\, \mathrm{Z}_{t+1}, \mathrm{Z}_{t+2}, \ldots, \mathrm{Z}_{t+k-1}) \,.$$

The main idea in computing it is that the pacf of Z_t and Z_{t+k} is the ACF of $Z_t - \hat{Z}_t$ and $Z_{t+k} - \hat{Z}_{t+k}$ where $\hat{Z}_{t+k}$ is the minimum mean square estimate (MMSE) of Z_{t+k} from the intermediate values. For the Gaussian process $\hat{Z}_t$ and $\hat{Z}_t$ are the conditional means. If we instead regress Z_{t+k} on all the values, the regression coefficient of Z_t will be the partial autocorrelation. This gives $Z_{t+k} = \hat{Z}_{t+k} + e_{t+k}$ or

$$\mathrm{Z}_{t+k} = \phi_{k,1}\mathrm{Z}_{t+k-1} + \phi_{k,2}\mathrm{Z}_{t+k-2} + \cdots + \phi_{k,k}\mathrm{Z}_t + e_{t+k} \tag{6}$$

where $\phi_{k,i}$ are the regression coefficients, and e_{t+k} is normal to $Z_t, \ldots, Z_{t+k-1}$, with e_{t+k} $N(0, v_k)$. If we multiply both sides with Z_{t+k-j}, take expectation and divide by γ_0 we get the expression

$$\rho_j = \phi_{k,1}\rho_{j-1} + \phi_{k,2}\rho_{j-2} + \cdots + \phi_{k,k}\rho_{j-k}$$

and could use Cramers rule to express $\phi_{k,k}$ as a ratio between determinants. This computation of $\phi_{k,k}$ will take $O(k^3)$ time. For computation in $O(k^2)$ time we can instead use the Durbin-Levinson recursions

$$\begin{aligned}\phi_{k+1,k+1} &= \frac{(\rho_{k+1} - \sum_{j=1}^{k}\phi_{k,j}\rho_{k+1-j})}{1 - \sum_{j=1}^{k}\phi_{k,j})\rho_j} \\ \phi_{k,j} &= \phi_{k,j} - \phi_{k+1,k+1}\phi_{k+1,k+1-j} \\ v_k &= v_{k-1}(1 - \phi_{kk}^2) \,.\end{aligned}$$

The function below named **acf2pacf** carry out this computation.

```r
acf2pacf <- function(rhoVec) {
    nextPhiVec = rep(NA, length(rhoVec))
    pacfVec = rep(NA, length(rhoVec))
    pacfVec[1] = rhoVec[1]
    phiVec = pacfVec
    vk = 35.29177777778 * (1 - (pacfVec[1])^2)
    for (k in 1:(length(rhoVec) - 1)) {
        sum1 = 0
        sum2 = 0
        for (j in 1:k) {
            # kloop
            sum1 = sum1 + phiVec[j] * rhoVec[k + 1 - j]
            sum2 = sum2 + phiVec[j] * rhoVec[j]
        }
        pacfVec[k + 1] = (rhoVec[k + 1] - sum1)/(1 - sum2)
        nextPhiVec[k + 1] = pacfVec[k + 1]
        vk[k + 1] = vk[k] * (1 - (pacfVec[k + 1])^2)
        for (j in 1:k) {
            # For j in 1:k
            nextPhiVec[j] = phiVec[j] - pacfVec[k + 1] * phiVec[k + 1 - j]
        }
        phiVec = nextPhiVec
    }
    return(list(weights = phiVec, variance = vk, pacf = pacfVec))
}
```

Problem 2

Now the R-function **arima.sim** was used to simulate $\{z_t\}_{t=1}^{1000}$ from the model. Figure 1 shows the simulated values for the close up $\{z_t\}_{t=200}^{300}$. The values seems to fluctuate around the mean 0, which is as expected because of the stationarity of the model.

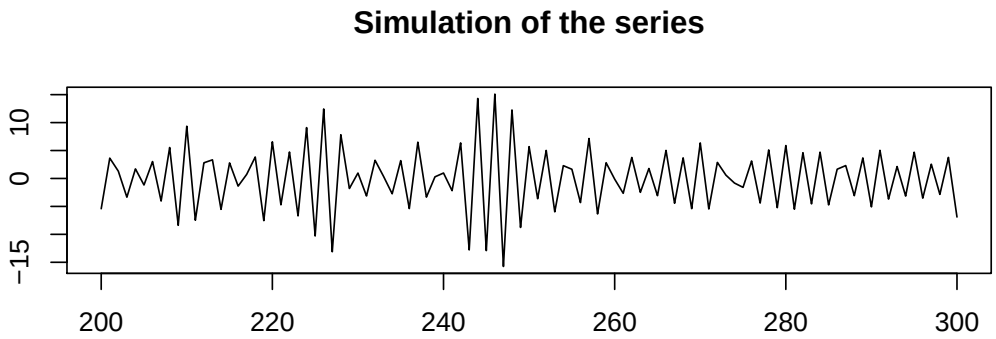

Figure 1: Simulated values for the close up $\{z_t\}_{t=200}^{300}$

The simulated values was used to estimate the autocorrelation function and the partial autocorrelation

function using the built-in functions **acf** and **pacf** in R. The results were plotted in Figure 2 and Figure 3 together with the theoretical values of the functions. The theoretical values of the autocorrelation function was found using (5) and the theoretical values of the partial autocorrelation function was found using the R function **acf2pacf** made in part 1. The theoretical values are plotted as red circles. From the figures it can be seen that the theoretical and estimated values are quite similar.

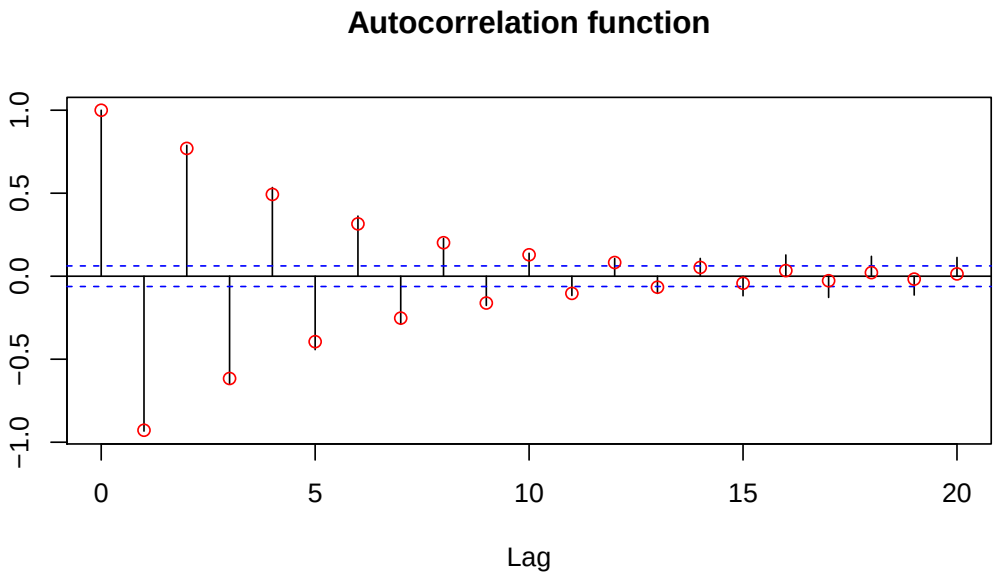

Figure 2: Theoretical and estimated autocorrelation function. The estimated function is shown in black and the theoretical values are shown as red circles.

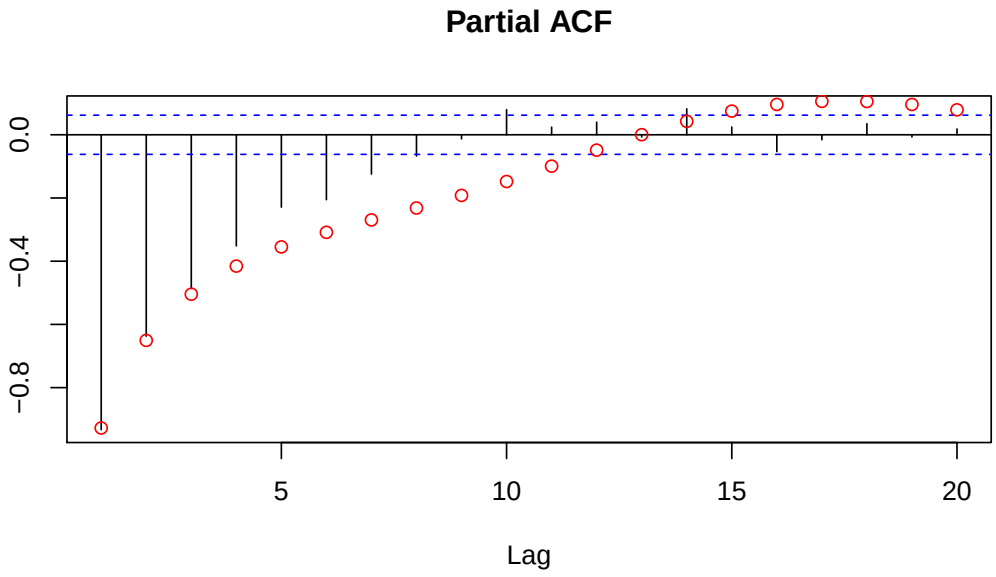

Figure 3: Theoretical and estimated partial autocorrelation function. The estimated function is shown in black and the theoretical values are shown as red circles.

To study the bias and the variance of the estimated acf and pacf, 1000 simulations was made of the series $\{z_t\}_{t=1}^{1000}$ and estimates of the acf and pacf was found for each of these simulations. The mean of these estimates are plotted together with the theoretical values in Figure 4 and Figure 5. The sample variance of the estimates was also found and from these estimates 95% limits was made and also plotted in the figures. From the plot of the autocorrelation function (Figure 4) the estimate looks slightly biased. It seems like the the acf is underestimated, that it obtains a smaller value than the true acf.

The estimates of the partial autocorrelation function in Figure 5 differs quite much from the true values, especially for higher lags. The estimates tends to be closer to zero than the true values and hence the estimate of the pacf also seems to be biased.

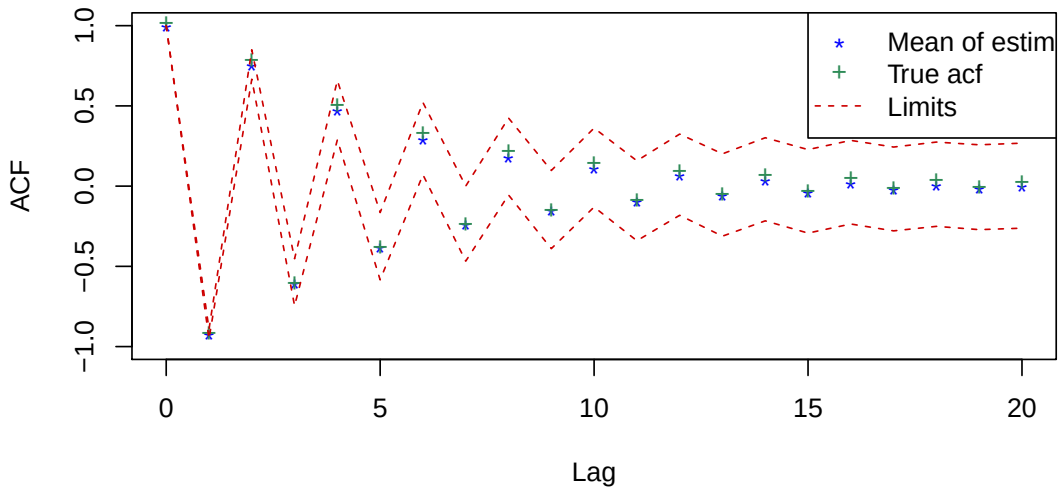

Figure 4: Mean of estimated ACF after 1000 simulations of the series $\{z_t\}_{t=1}^{1000}$, true ACF and 95 % limits for the estimated ACF.

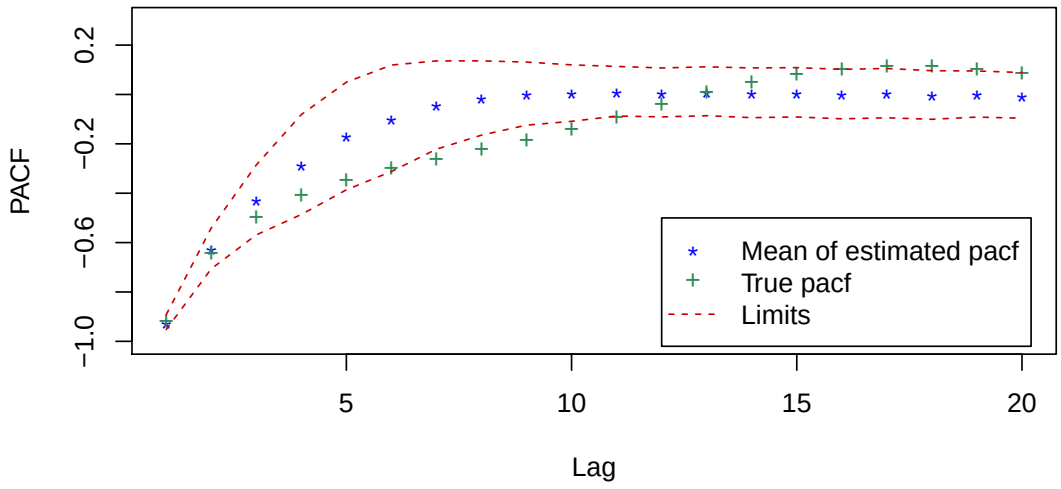

Figure 5: Mean of estimated PACF after 1000 simulations of the series $\{z_t\}_{t=1}^{1000}$, true PACF and 95 % limits for the estimated PACF.

Then the same is done for a longer time series $\{z_t\}_{t=1}^{30000}$, see Figure 6 and 7. In this case the bias is much smaller. The estimates of the autocorrelation function seems unbiased, and the bias in the pacf is much smaller. This is because the sample acf and pacf is asymptotically unbiased. The variance obtained is also much smaller for the long time series, than for a smaller time series.

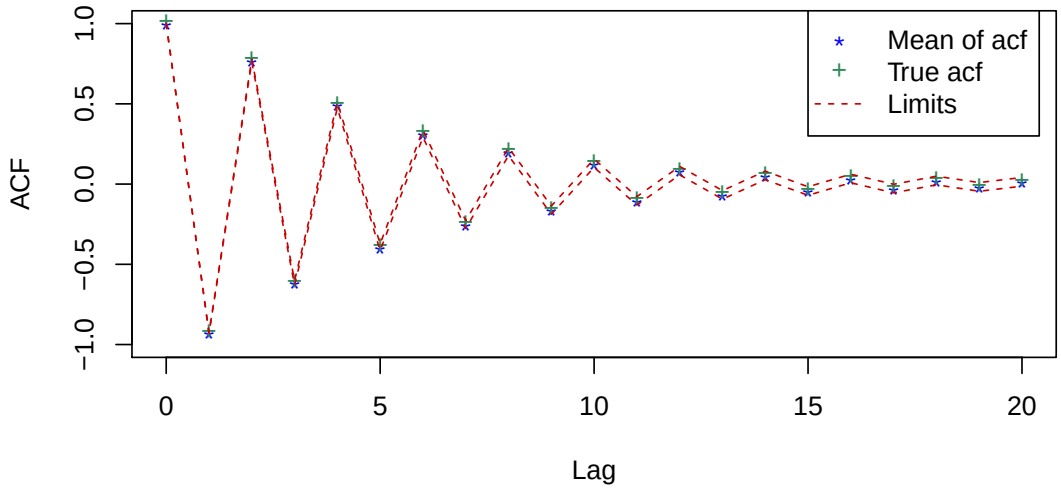

Figure 6: Mean of estimated ACF after 1000 simulations of the series $\{z_t\}_{t=1}^{30000}$, true ACF and 95 % limits for the estimated ACF.

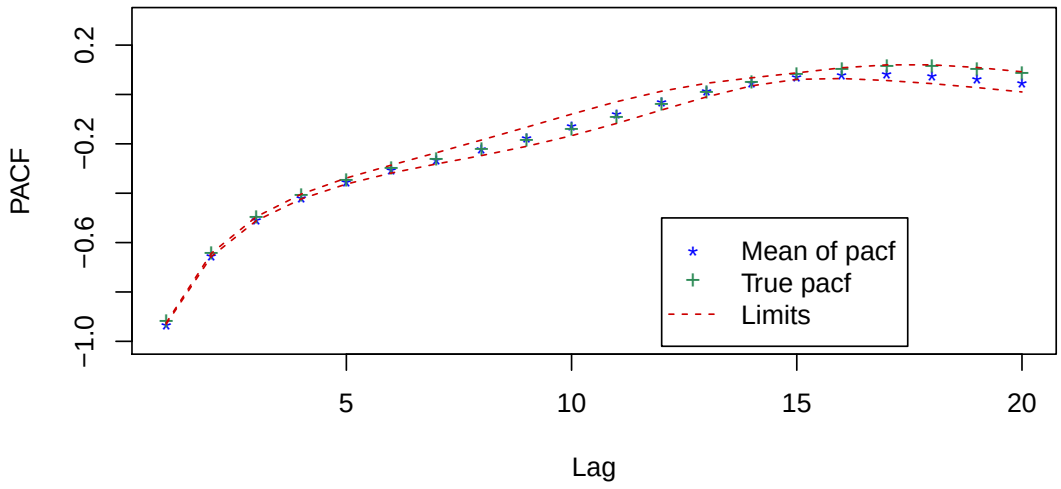

Figure 7: Mean of estimated PACF after 1000 simulations of the series $\{z_t\}_{t=1}^{30000}$, true PACF and 95 % limits for the estimated PACF.

Problem 3

Now we want to implement our own code simulating from the model. First an initial value for Z_1 must be drawn. This is done by drawing a value from the distribution $p(Z_1|a_0, a_0)$. Since $Z_1 \sim N(0, \gamma_0)$ and $a_0 \sim a_1 \sim N(0, \sigma_a^2)$. the joint distribution is given by

$$\begin{bmatrix} Z_1 \\ a_0 \\ a_1 \end{bmatrix} \sim N\left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \gamma_0 & \sigma_a^2(\phi_1 - \theta_1) & \sigma_a^2 \\ \sigma_a^2(\phi_1 - \theta_1) & \sigma_a^2 & 0 \\ \sigma_a^2 & 0 & \sigma_a^2 \end{bmatrix} \right)$$

It is known that if $\boldsymbol{X} = \begin{pmatrix} \boldsymbol{X_1} \\ \boldsymbol{X_2} \end{pmatrix}$ is multinormal with expectation $\boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu_1} \\ \boldsymbol{\mu_2} \end{pmatrix}$ and covariance matrix $\boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma_{11}} & \boldsymbol{\Sigma_{12}} \\ \boldsymbol{\Sigma_{21}} & \boldsymbol{\Sigma_{22}} \end{pmatrix}$ then the conditional distribution for $\boldsymbol{X_1}$ given $\boldsymbol{X_2} = \boldsymbol{x_2}$ is also multinormal with expectation and covariance matrix given by

$$\begin{aligned} E(\boldsymbol{X_1}|\boldsymbol{X_2} = \boldsymbol{x_2}) &= \boldsymbol{\mu_1} + \boldsymbol{\Sigma_{12}}\boldsymbol{\Sigma_{22}^{-1}}(\boldsymbol{x_2} - \boldsymbol{\mu_2}) \\ \text{Cov}(\boldsymbol{X_1}|\boldsymbol{X_2} = \boldsymbol{x_2}) &= \boldsymbol{\Sigma_{11}} - \boldsymbol{\Sigma_{12}}\boldsymbol{\Sigma_{22}^{-1}}\boldsymbol{\Sigma_{21}} \end{aligned} \tag{7}$$

Now, from the joint distribution and the formulas (7) the mean and variance of the conditional distribution of $[Z_1|a_1, a_0]$ are found

$$\begin{aligned} E(Z_1|a_1, a_0) &= (\phi_1 - \theta_1)a_0 + a_1 \\ \text{Cov}(Z_1|a_1, a_0) &= \gamma_0 - \sigma_a^2(\phi_1 - \theta_1)^2 \end{aligned} \tag{8}$$

Then to simulate from the ARMA-model first Z_1 is drawn from the Gaussian distribution with the parameters given from (8) and then Z_j is found with

$$Z_j = \phi_1 Z_{j-1} + a_t - \theta_1 a_{t-1} - \theta_2 a_{t-2}$$

where a_t is drawn from the Gaussian distribution $a_t \sim N(0, \sigma_a^2)$.

The code of the function is shown below

```
SimulateARMA = function(N, phi, theta1, theta2, sigma) {

    # Finding Var(Z)
    gamma0 = sigma/(1 - phi1^2) * (1 - 2 * phi1 * theta1 - 2 * phi1 * theta2 *
        (phi1 - theta1) + theta1^2 + theta2^2)

    # Creating the vector containing the random noise a_t for all t=0...N
    noise = rnorm(N + 1, 0, sigma)

    # Computing the parameters for the initial distribution
    mu0 = c(sigma * (phi1 - theta1), sigma) %*% (1/sigma * diag(2)) %*% t(t(c(noise[1],
        noise[2])))
    var0 = gamma0 - c(sigma * (phi1 - theta1), sigma) %*% (1/sigma * diag(2)) %*%
        t(t(c(sigma * (phi1 - theta1), sigma)))

    # Initializes Z (the vector containing the simulated values)
    Z = rep(0, times = N)

    # Draws the start value from the joint distribution of Z1 a0 and a1.
    Z[1] = rnorm(1, mu0, sqrt(var0))

    for (i in 2:N) {
        Z[i] = phi1 * Z[i - 1] + noise[i + 1] - theta1 * noise[i] - theta2 *
            noise[i - 1]
    }
    return(Z)
}
```

The function **SimualteARMA** was then used to simulate $\{z_t\}_{t=1}^{1000}$ from the model. Figure 8 shows the simulated values for the close up $\{z_t\}_{t=200}^{300}$.

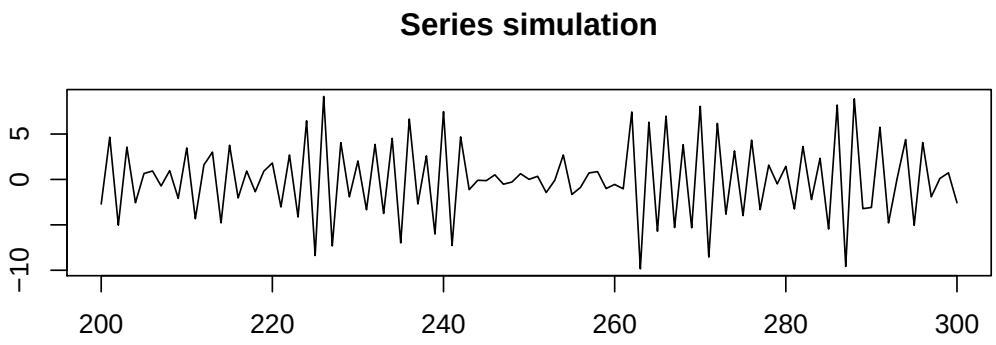

Figure 8: Simulated values for the close up $\{z_t\}_{t=200}^{300}$

The simulated values was used to estimate the autocorrelation function and the partial autocorrelation function using the built-in functions **acf** and **pacf** in R. The results was plotted in Figure 9 and Figure 10 together with the theoretical values of the functions (red circles). The estimated values coincide quite well with the true values, indicating that the implementation is correct.

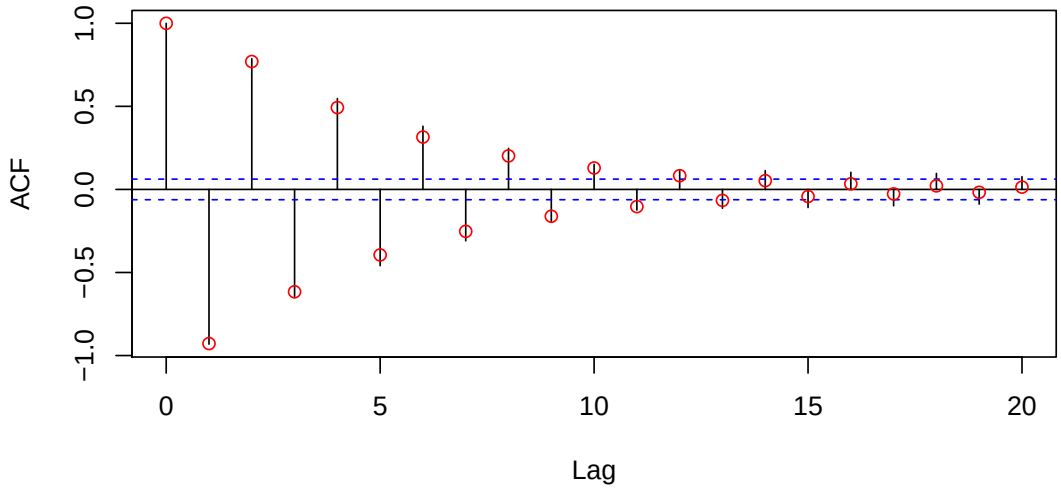

Figure 9: Theoretical and estimated autocorrelation function. The estimated function is shown in black and the theoretical values are shown as red circles.

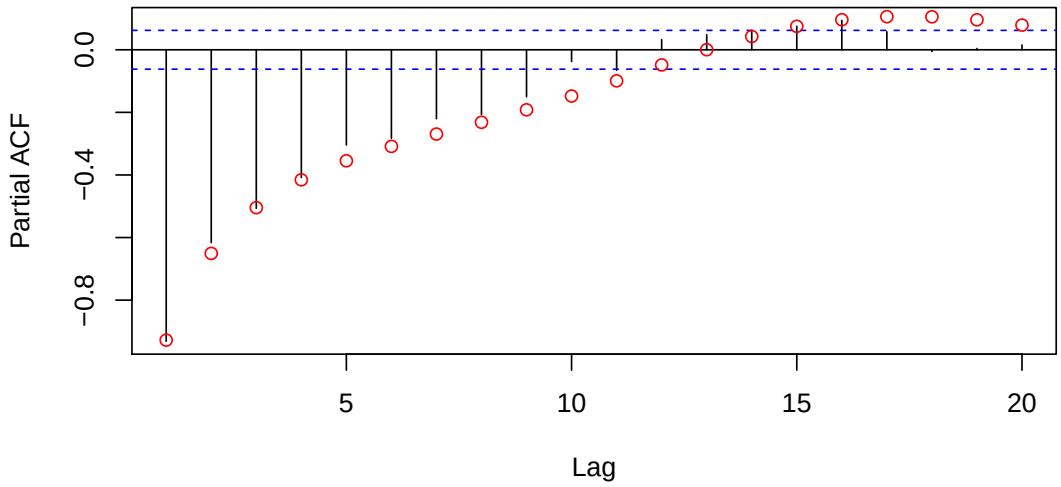

Figure 10: Theoretical and estimated partial autocorrelation function. The estimated function is shown in black and the theoretical values are shown as red circles.

To check that the implementation gives a result that is stationary in variance, 500 simulations of the series $\{z_t\}_{t=1}^{1000}$ was run and the variance in each timestep t was computed. The values are plotted in Figure 11. The variance seems to fluctuate randomly around a center value close to the theoretical variance (4). This indicates that the time series is stationary in variance and that the initial condition used in our implementation is correct.

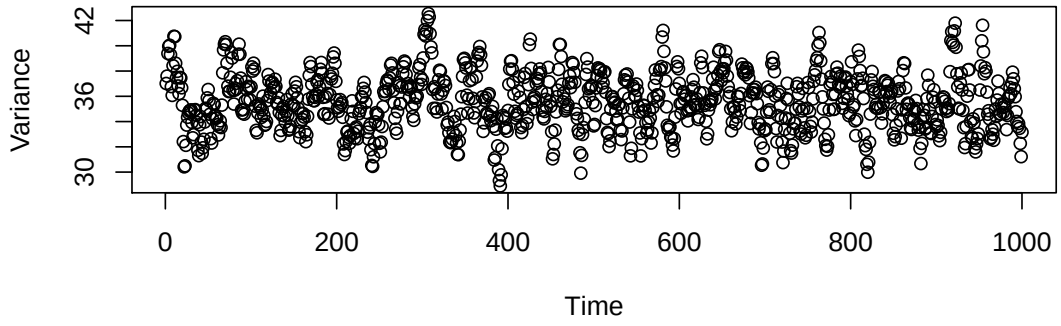

Figure 11: Variance in each timestep for the 500 simulations.

Problem 4

Now we will discuss the theoretical pacf of our model. The partial autocorrelation function of an AR(p) model cuts off after the pth value, while it tails off for a MA(q) model. The combined ARMA(p,q) model will in the beginning be influenced both by the AR and the MA part, but in its tail it will behave similary to a pure MA(q) model. As the PACF of an MA(2) process with complex roots will tail off as a damped sine wave, our model will exhibit this behaviour in its tail.

The auto covariances of an AR(2) process with weights corresponding to our θ(B) is

$$\mathrm{E}(Z_{t-k}Z_t) = \theta_1\mathrm{E}(Z_{t-k}Z_{t-1} + \theta_2\mathrm{E}(Z_{t-k}Z_{t-2} + \mathrm{E}(Z_{t-k}a_t)$$

which gives

$$\rho_k = \theta_1\rho_{k-1} + \theta_2\rho_{k-2}\ , k \geq 1 \rho_k = \theta_1\mathrm{B}\rho_k + \theta_2\mathrm{B}^2\rho_k 0 = (1 - \theta_1 textB - \theta_2 textB^2)\rho_k\ .$$

For complex roots the general real solution is on the form

$$\rho_j = \alpha^j(b_1 cos\omega t + b_2 cos\omega t)\ . \tag{9}$$

From equation (1) we see that in our case $\alpha = 0.99$ and $\omega = 0.319$, which will be similar to the behavior of the partial acf of the MA(2) process with the same parameters due to the duality between these. Figure [12] shows the theoretical partial acf values for the full ARMA(1,2) model along with and the period and the exponential decay equation (9). As b_1 and b_2 is yet to be determined we adjust the lines to the graph by reading off a value of the partial acf at maximal amplitude. The first values of the partial acf are more extreme, but already around the fifteenth value the sinusoidal shape is visible. It is clear from the graph that the behavior indeed is like that of a damped sine wave. The values for the dampening and period from (9) fit the partial acf closely but not perfectly. This is precisely what we would expect. We would not expect the fit to be perfect as the behavior of the pacf is similar to that of the corresponding acf but not exactly the same.

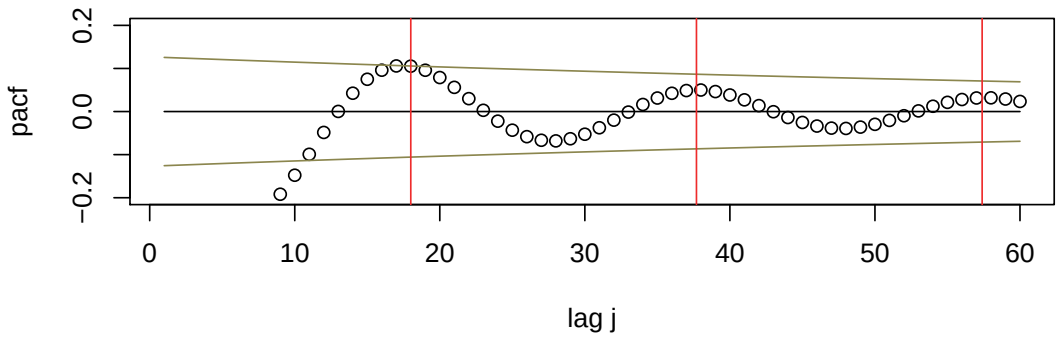

Figure 12: Partial autocorrelation function of lags up to 60. The red lines mark the period and the grey lines mark the dampening.

Problem 5

The AR(∞) representation of our model is given by

$$\pi(\mathrm{B})Z_t = a_t$$

where

$$\pi(\mathrm{B}) = 1 - \sum_{j=1}^{\infty} \pi_j \mathrm{B}^j = \frac{\phi(\mathrm{B})}{\theta(\mathrm{B})}$$

Now the forecast is done by computing a weighted average of the previous observations. In our case this leads to

$$(1 - \phi_1 B) = (1 - \pi_1 B - \pi_2 B^2 - \pi_2 B^2 - \cdots)(1 - \theta_1 B - \theta_2 B^2)$$

Then rearranging

$$(1 - \phi_1 \mathrm{B}) = 1 - (\pi_1 + \theta_1)\mathrm{B} - (\pi_2 - \pi_1\theta_1 + \theta_2)\mathrm{B}^2 - (\pi_3 + \pi_2\theta_1 + \pi_1\theta_2)\mathrm{B}^3 - (\pi_4 + \pi_3\theta_2 + \pi_2\theta_3) - \cdots$$

Collecting terms with the same power of B gives

$$\begin{aligned} \mathrm{B}^0 &: 1 = 1 \\ \mathrm{B}^1 &: \pi_1 = \phi_1 - \theta_1 \\ \mathrm{B}^2 &: \pi_2 = \pi_1\theta_1 - \theta_2 \\ \mathrm{B}^j, j \geq 3 &: \pi_j = \pi_{j-1}\theta_1 + \pi_{j-2}\theta_2 \end{aligned}$$

These weights are found in R and are plotted in Figure 13.

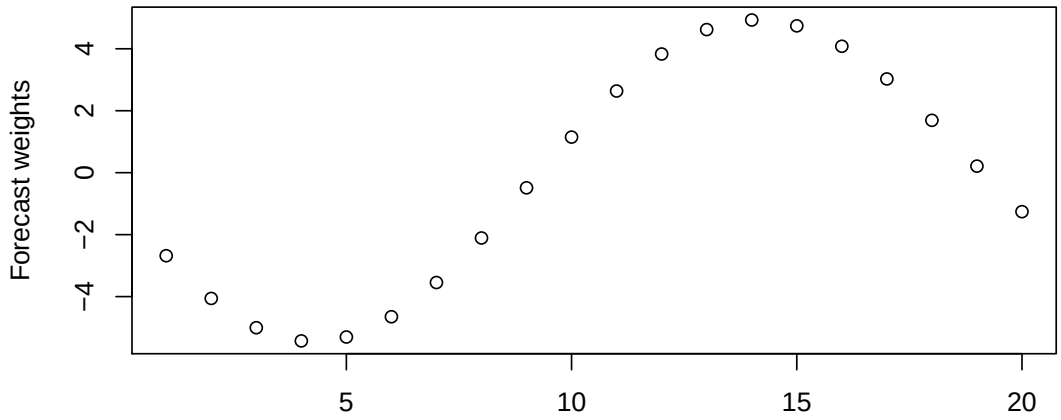

Figure 13: The forecast weights

From these weights the l-step ahead forecast can be found by

$$\hat{Z}_n(l) = \sum_{j=1}^{\infty} \pi_j \hat{Z}_n(l-j), \quad l \geq 1 \tag{10}$$

The forecast error variance is given by

$$Var(e_n(l)) = Var(Z_{n+l} - \hat{Z}_n(l)) = \sigma_a^2 \sum_{j=0}^{l-j} \psi_j^2$$

where ϕ_j are the jth coefficient if the model were written in a MA(∞) representation. These coefficients are in our case given by

$$\begin{aligned}\psi_0 &= 1\\ \psi_1 &= \phi_1 - \theta_1\\ \psi_2 &= \phi_1^2 - \theta_1\phi_1 - \theta_2\\ \psi_j &= \psi_2\phi_1^{j-2}\end{aligned}$$

The time series used for forecasting is downloaded with the command **load(url("https://www.math.ntnu.no/ jarlet/tidsrekker/zforecast.Rdata"))** in R and it is plotted in Figure 14.

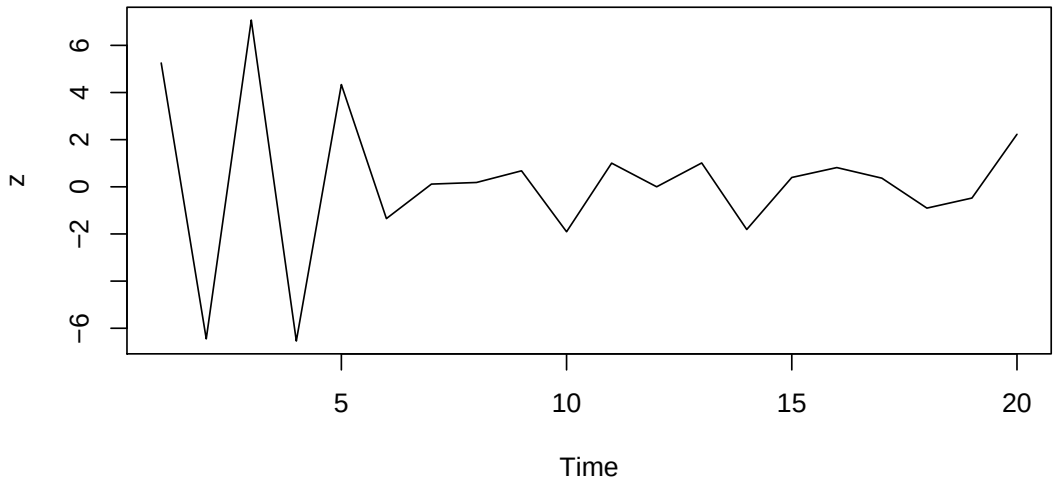

Figure 14: Time series used for forecasting

Since $Z_0, Z_{-1}, \ldots$ are not available, an approximated forecast is computed based on the known values in this time series and hence dropping the terms containing $Z_0, Z_{-1}, \ldots$. Implementation of the method in R gives the 1-step ahead forecast

$$\hat{Z}_n(1) = -7.75 \qquad \mathrm{Var}(e_n(1)) = \sigma_a^2 = 1$$

Problem 6

Equation (10) was used to compute the l-step ahead forecast of the time series. The code is shown below and the result is plotted in Figure 15 together with 95% forecast limits for the steps l=$1, 2, \ldots, 10$.

```
# Initializing the parameters
phi1 = -0.8
theta = c(1.88, -0.98)
```

```r
sigma = 1

# Loading the data and plotting the data. (Here the data is stored in the
# file z.RData)
load("z.RData")
plot(z, xlim = c(0, 30), ylim = c(-27, 27))
t = length(z)


# Computing the coefficients of the AR representation
pi = phi1 - theta[1]
pi[2] = theta[1] * pi[1] - theta[2]

for (j in 3:t) {
    pi[j] = theta[1] * pi[j - 1] + theta[2] * pi[j - 2]
}

# Computing the forecasts
zhat = sum(pi * z[20:1])
zhat[2] = pi[1] * zhat[1] + sum(pi[2:20] * z[20:2])

ll = 3:10
zhat[ll] = phi1^(ll - 2) * zhat[2]

# Plotting the forecast
points(20 + 1:10, zhat, pch = "+")

# Then computing the coefficients of the MA representation to obtain the
# variance
psi = phi1 - theta[1]
psi[2] = phi1^2 - theta[1] * phi1 - theta[2]
ii = 3:9
psi[ii] = psi[2] * phi1^(ii - 2)

# Finding the variance given the MA representation
variance = sigma * (cumsum(c(1, psi^2)))

# Plotting the forecast limits
lines(20 + (1:10), zhat + qnorm(0.025) * sqrt(variance), col = "red3", lty = 2)
lines(20 + (1:10), zhat + qnorm(0.975) * sqrt(variance), col = "red3", lty = 2)
```

Table 1: The 1 step forecasts and their forecast error variances.

	Forecast	Variance
Problem 6	-7.754	1
Problem 7	-3.449	1.0705

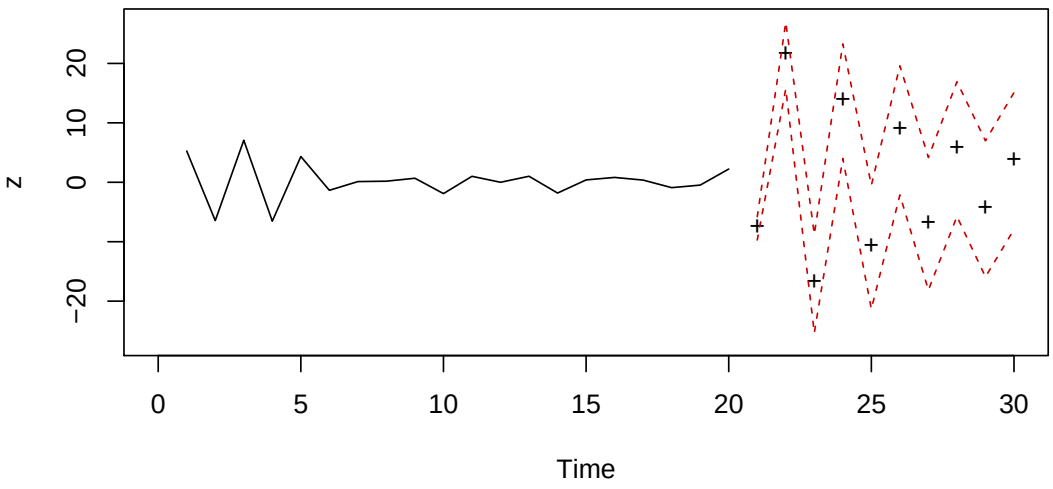

Figure 15: The forecasted values together with 95% forecast limits

From Figure 15 it can be seen that the error variance gets higher for increasing l which is as expected. The variance seems to go towards a fixed value for large l which is also as expected since our model is stationary. Also the forecasted values seems to go towards the mean 0 for large l.

Problem 7

In this exercise we use the regression in equation (6) to compute the optimal 1 step forecast in our ARMA$(1, 2)$ model. This will be the optimal forecast as we have finite observations. For a stationary Gaussian process the autocovariance, autocorrelation and partial autocorrelation are independent of the time index, so when the model is known the forecast weights $\phi_{k,i}$ are also known. Thus the 1 step forecast and its error variance can be computed from a finite number of successive known states. The R function in problem 1 **acf2pacf** computes k weights and the following forecast error variance for our model. Multiplying the weights with the observed values we obtain the forecast.

The 1 step forecast from problem 6 and 7 are displayed together in table [1]. As we can see the forecast error variance for the method in problem 6 is estimated to be lower than that in problem 7. This will be discussed further in problem 8. The forecast weights of both methods are displayed in figure [16]. The weights for the optimal forecast are of lower absolute value than those for the forecast from problem 6. The difference in weights is very large compared to the size of the weights which makes the forecasts quite different. This is also discussed more in detail in problem 8.

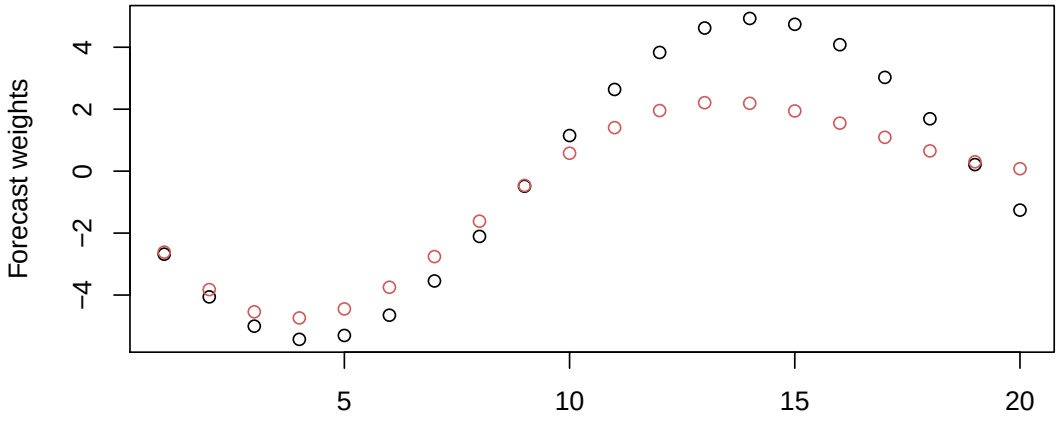

Figure 16: Forecast weights from problem 6 in black, and from problem 7 in red.

Problem 8

In problem 6 and 7 we found the one step forecast with two different methods. The method in promblem 6 computed the forecast in $O(n)$ time, while the method in promlem 7 uses $O(n^2)$ time to find the weights. The estimate for the forecast error variance is only 1 in problem 6, but 1.07 in problem 7. This does not mean that the accuracy is better in problem 6. In problem 7 we made the optimal forecast based on the observations we actually had, while the forecast in problem 6 would only have been optimal had we known all previous values. As such the accuracy is better in problem 7.

The two methods yield forecast values -7.75 and $?? - 3.45$, which are very different compared to the total variability of values in this model. These values are so different because the forecast weights are so different. The forecast from problem 7 is the optimal forecast, and the one estimate for the forecast that should be trusted. The forecasts are so different because the forecast in problem 6 is quite lacking. To obtain it we expressed the model in AR(∞) form, and dropped the weights corresponding to unknown values. As the roots of the MA-polynomial are very close to the unit circle, with size of roots approximately 1.01, the weights will remain sizeable for quite a lot of observations. In the data we had there were only 20 observations, and the weights before these were dropped as the method requires. As the weights were still quite sizeable this resulted in a bad approximation. More observations were needed in order to make a good forecast using method 6.

If we were to try and compute the forecast for a non-invertible, stationary ARMA-model with the method from problem 6 we would see that the size of the weights would not decrease when going backwards in time. If the roots of the MA-polynomial were on the unit circle, the size of the weights would simply not decrease, but if the roots were inside the unit circle the weights would increase exponentially backwards in time. The most important information would lie in the weights infinitely prior to the observed values, and dropping the weights before observed values would render any attempt at a forecast completely useless. The method from problem 7 would work for a non-invertible stationary ARMA model as stationarity only is required. However for an invertible non-stationary ARMA model the situation would be opposite, with only the method from 6 being able to make a forecast.