Simulation of AR(p) and Seasonal processes

Assignment 2 – 02417 Time Series Analysis – Anders Hørsted (s082382)

This report consists of four parts. In the first part a few theoretical results for the AR(2) process are obtained and a prediction is made for a seasonal model. In part two a few different techniques for simulation of AR-processes are tested and the results from using different coefficients in the lag polynomial are compared and commented on. In part three the random walk process is simulated and plotted along with the autocorrelation function of the process. Finally in part four a couple of seasonal models are simulated and the autocorrelation function is plotted.

Part 1: Theory and prediction of seasonal model

The AR(2)-process

We consider the AR(2)-process

$$X_t + \phi_1 X_{t-1} + \phi_2 X_{t-2} = \varepsilon_t$$

where ε_t is white noise. First the parameters for which the AR(2)-process is stationary are determined.

From theorem 5.9 in [1] the roots of the equation $\phi(z^{-1}) = 0$, with respect to z, should all lie within the unit circle for the AR(2)-process to be stationary. Now assuming $z \neq 0$ we get

$$\phi(z^{-1}) = \phi_2 z^{-2} + \phi_1 z^{-1} + 1$$

so the roots are found from

$$\begin{aligned} \phi_2 z^{-2} + \phi_1 z^{-1} + 1 = 0 \quad &\Leftrightarrow \\ z^2 + \phi_1 z + \phi_2 = 0 \end{aligned}$$

With $D = \phi_1^2 - 4\phi_2$ we find the roots as

$$z_1 = \frac{-\phi_1 + \sqrt{D}}{2}, \quad z_2 = \frac{-\phi_1 - \sqrt{D}}{2}$$

We now look at the three cases $D = 0, D < 0, D > 0$ one at a time.

First $D = 0$. Then we get a real double root given by $z_1 = z_2 = -\frac{\phi_1}{2}$ and since this root should be within the unit circle, we get the inequality

$$\frac{|\phi_1|}{2} < 1 \quad \Leftrightarrow \quad \phi_1 \in (-2, 2)$$

Since $D = 0 \Leftrightarrow \phi_1^2 = 4\phi_2$ we get the first set

$$M_1 = \{(\phi_1, \phi_2) \,|\, \phi_1^2 = 4\phi_2, \phi_1 \in (-2, 2)\}$$

for which the AR(2)-process is stationary

For the second case $D < 0$ we get two complex roots given by

$$z_1 = \frac{-\phi_1}{2} + \frac{\sqrt{4\phi_2 - \phi_1^2}}{2}\, i, \quad z_2 \frac{-\phi_1}{2} - \frac{\sqrt{4\phi_2 - \phi_1^2}}{2}\, i$$

where both roots have the same modulus $|z_1| = |z_2| = \sqrt{\phi_2}$. Therefore to get stationarity $\sqrt{\phi_2} < 1 \Leftrightarrow \phi_2 \in [0, 1)$, which combined with $D < 0 \Leftrightarrow \phi_1^2 < 4\phi_2$ gives $\phi_1 \in (-2, 2)$ and the second parameter set becomes

$$M2 = \{(\phi_1, \phi_2) \,|\, \phi_1^2 < 4\phi_2, \phi_1 \in (-2, 2)\}$$

For the last case $D > 0$ the root of interest depends on the sign of ϕ_1. If $\phi_1 < 0$ we look at z_1 and for $\phi_1 > 0$ we look at z_2. Both cases can be handled by one inequality given by

$$\begin{aligned} &\frac{|\phi_1| + \sqrt{\phi_1^2 - 4\phi_2}}{2} < 1 \quad \Leftrightarrow \\ &|\phi_1| + \sqrt{\phi_1^2 - 4\phi_2} < 2 \end{aligned}$$

From this we get that $\phi_1 \in (-2, 2)$. We now look at all lines for which $\phi_2 = |\phi_1| - 1$, giving

$$\begin{aligned} |\phi_1| + \sqrt{\phi_1^2 - 4\phi_2} &= |\phi_1| + \sqrt{\phi_1^2 - 4(|\phi_1| - 1)} \\ &= |\phi_1| + \sqrt{(|\phi_1| - 2)^2} \\ &= |\phi_1| + |\,|\phi_1| - 2\,| \\ &= 2 \end{aligned}$$

using that for $\phi_1 \in (-2, 2)$, $|\,|\phi_1| - 2\,| = 2 - |\phi_1|$. From the expression for D we conclude that $\phi_2 > |\phi_1| - 1$. Combined with $D > 0 \Leftrightarrow \frac{1}{4}\phi_1^2 > \phi_2$ the last set is

$$M_3 = \{(\phi_1, \phi_2) \,|\, |\phi_1| - 1 < \phi_2 < \frac{1}{4}\phi_1^2, \phi_1 \in (-2, 2)\}$$

The final result is that the AR(2)-process is stationary for parameters in the set

$$M = M_1 \cup M_2 \cup M_3 = \{(\phi_1, \phi_2) \,|\, |\phi_1| - 1 < \phi_2 < 1\}$$

The set M is sketched in figure 1.

We now need to determined the set of parameters for which the autocorrelation function of the AR(2)-process shows damping harmonic oscillations. Based on equation 5.83

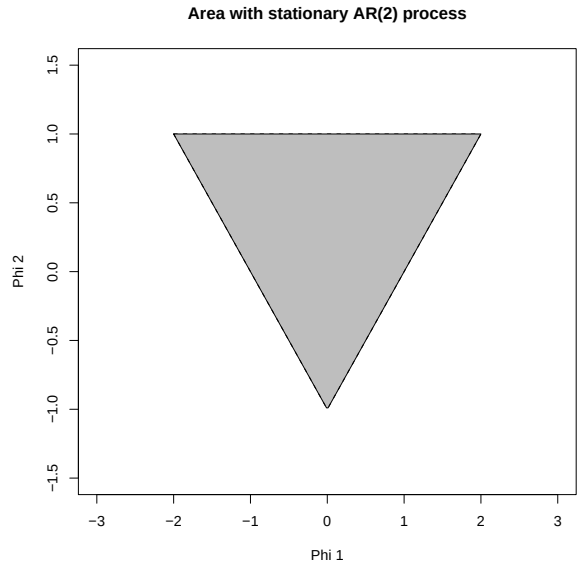

Figure 1: *Set of the (ϕ_1, ϕ_2)-plane where the characteristic equation for an AR(2) process has roots within the unit circle.*

and the surrounding text in [1] the autocorrelation function shows damping harmonic oscillations exactly when the roots of the characteristic equation are complex. Using the result from the previous paragraphs the set where the autocorrelation function shows damping harmonic oscillations is given by

$$M2 = \{(\phi_1, \phi_2) \,|\, \phi_1^2 < 4\phi_2, \phi_1 \in (-2, 2)\}$$

The set $M2$ is sketched in figure 2

Prediction in seasonal model

We now model the deviation Y_t between observed and calculated water levels as a linear model given by

$$(1 - 0.8B)(1 - 0.2B^6)(1 - B)Y_t = \varepsilon_t \tag{1}$$

where ε_t is white noise with variance σ_ε^2. An estimate based on around 1500 observations is calculated as $\hat{\sigma}_\varepsilon^2 = 0.31\,\mathrm{dm}^2$. Based on the data $D = \{Y_1, Y_2, \ldots, Y_{10}\}$ shown in table 1 in the assignment paper, we will predict Y_{12}.

First the operator polynomial in model (1) is expanded giving

$$(1 - 1.8B + 0.8B^2 - 0.2B^6 + 0.36B^7 - 0.16B^8)Y_t = \varepsilon_t$$

that rewritten gives

$$Y_t = 1.8Y_{t-1} - 0.8Y_{t-2} + 0.2Y_{t-6} - 0.36Y_{t-7} + 0.16Y_{t-8} + \varepsilon_t$$

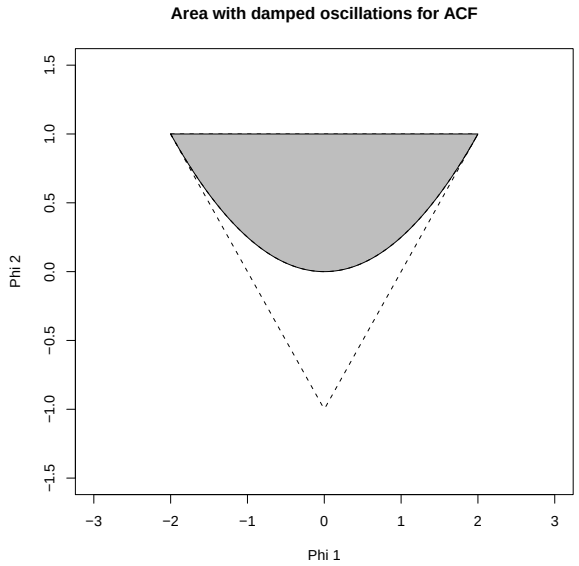

Figure 2: *Set of the (ϕ_1, ϕ_2)-plane where the autocorrelation function for an AR(2) process shows damping harmonic oscillations.*

We can now predict Y_t at time $t = 11$ by

$$\begin{aligned}\widehat{Y}_{11} &= \mathrm{E}[Y_{11} \mid D] \\ &= 1.8Y_{10} - 0.8Y_9 + 0.2Y_5 - 0.36Y_4 + 0.16Y_3 \\ &= -5.44\end{aligned}$$

The prediction $\widehat{Y}_{11}$ is then used to find $\widehat{Y}_{12}$

$$\begin{aligned}\widehat{Y}_{12} &= \mathrm{E}[Y_{12} \mid D] \\ &= 1.8\widehat{Y}_{11} - 0.8Y_{10} + 0.2Y_6 - 0.36Y_5 + 0.16Y_4 \\ &= -6.43\end{aligned}$$

To get a 95% confidence interval for $\widehat{Y}_{12}$ we need to find the variance of the prediction error $Y_{12} - \widehat{Y}_{12}$.

$$\begin{aligned}\mathrm{Var}[Y_{12} - \widehat{Y}_{12} \mid D] &= \mathrm{Var}[1.8Y_{11} + \varepsilon_{12} \mid D] \\ &= \mathrm{Var}[1.8(1.8Y_{10} - 0.8Y_9 + 0.2Y_5 - 0.36Y_4 + 0.16Y_3 + \varepsilon_{11}) + \varepsilon_{12} \mid D] \\ &= \mathrm{Var}[1.8\varepsilon_{11} + \varepsilon_{12}] \\ &= (1.8^2 + 1)\sigma_\varepsilon^2\end{aligned}$$

Since σ_ε^2 is unknown we use the estimate $\hat{\sigma}_\varepsilon^2$ instead which gives

$$\begin{aligned}\mathrm{Var}[Y_{12} - \widehat{Y}_{12} \mid D] &= (1.8^2 + 1)\hat{\sigma}_\varepsilon^2 \\ &= 1.31\end{aligned}$$

And finally we get the 95% confidence interval for $\widehat{Y}_{12}$ as

$$-6.43 \pm 1.96 \cdot \sqrt{1.31} = [-8.68; -4.18]$$

Part 2: Simulating AR-processes

In this part we will simulate a few different AR-processes using the `filter` and `arima.sim` functions in R. The AR(2)-process is written as

$$Y_t = a_1 Y_{t-1} + a_2 Y_{t-2} + e_t$$

Simulating the AR(1)-process

We first simulate the AR(1)-process for coefficients $a_1 = 0.1, -0.1, 0.9, -0.9$ and $a_2 = 0$. The results are shown in figure 3 and figure 4. From the figures it is seen that the ACF for the two models with numerically small a_1 decays rapidly to zero while the ACF for the models with numerically large a_1 decays slower. The model with numerically small a_1 is looking much like white noise, while the models with numerically large coefficients seems closer to a non-stationary process. The extreeme case where $|a_1| > 1$ is plotted in figure 5 and it is seen that the ACFs for these functions are almost constant which shows that these processes are indeed non-stationary. This corresponds with the fact that the root of the characteristic equation for an AR(1)-process is equal to $-a_1$, so for $|a_1| > 1$ the absolute value of the root will also be greater than one. From theorem 5.9 in [1] we get that the AR process is non-stationary for roots with abolute value greater than one which match the results in the plots.

Simulating the AR(2)-process

Next we will simulate an AR(2)-process. To make sure the process is stationary we choose two roots $-1 < r_1, r_2 < 1$ and then calculate

$$(x - r_1)(x - r_2) = x^2 - (r_1 + r_2)x + r_1 r_2$$

from which we see that we should choose $a_1 = r_1 + r_2$ and $a_2 = -r_1 r_2$. Two pairs of roots are now selected as $(-\frac{1}{2}, \frac{3}{4}), (\frac{2}{5}, \frac{9}{10})$ and the AR(2)-process is simulated, using the `filter` function, for these two models. The results are shown in figure 6. In the figure it seen that the ACF decays relatively fast for both models, which was expected since both models are staionary. The same models are simulated with the `arima.sim` function and the results are shown in figure 7. The same damping behaviour of the ACFs are observed although the ACF of the right model have values above the 95%-prediction interval for lags around 20-25. It could look like a damped sine function, which still corresponds nicely with a non-stationary function.

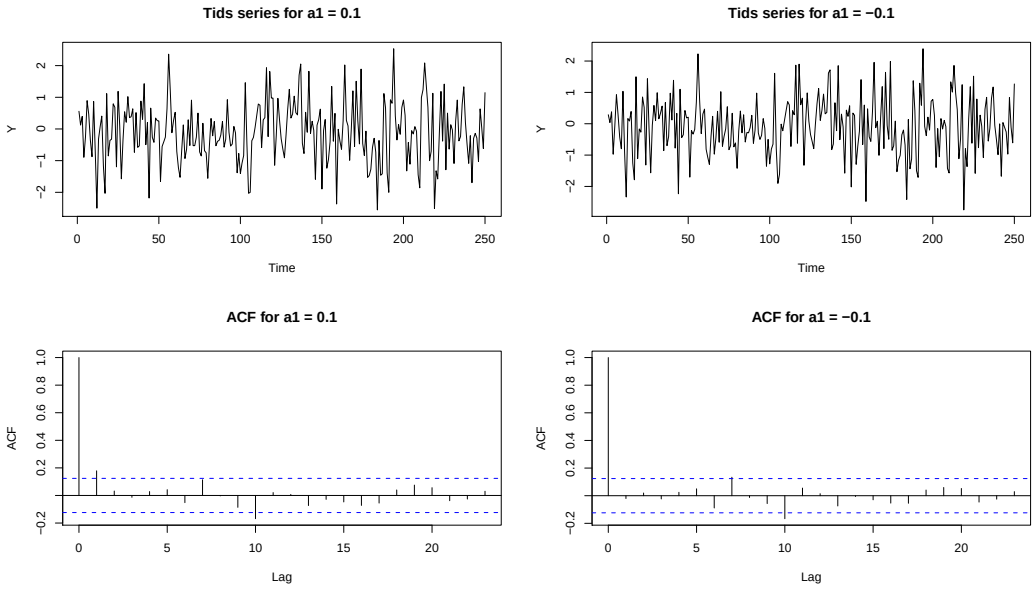

Figure 3: *Simulation of the AR(1)-process (with $a_1 = 0.1$ and $a_1 = -0.1$) using the* `filter` *function*

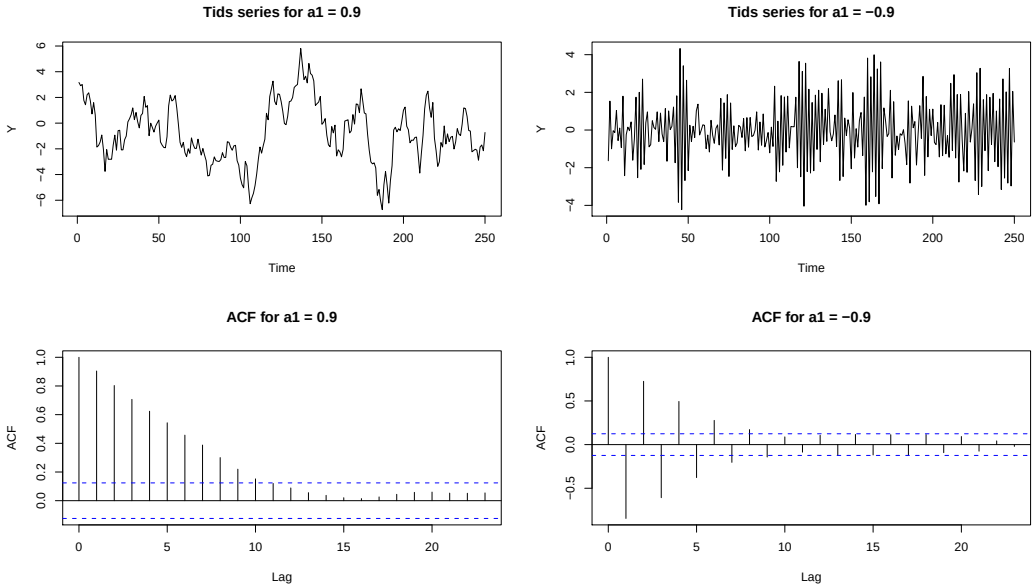

Figure 4: *Simulation of the AR(1)-process (with $a_1 = 0.9$ and $a_1 = -0.9$) using the* `filter` *function*

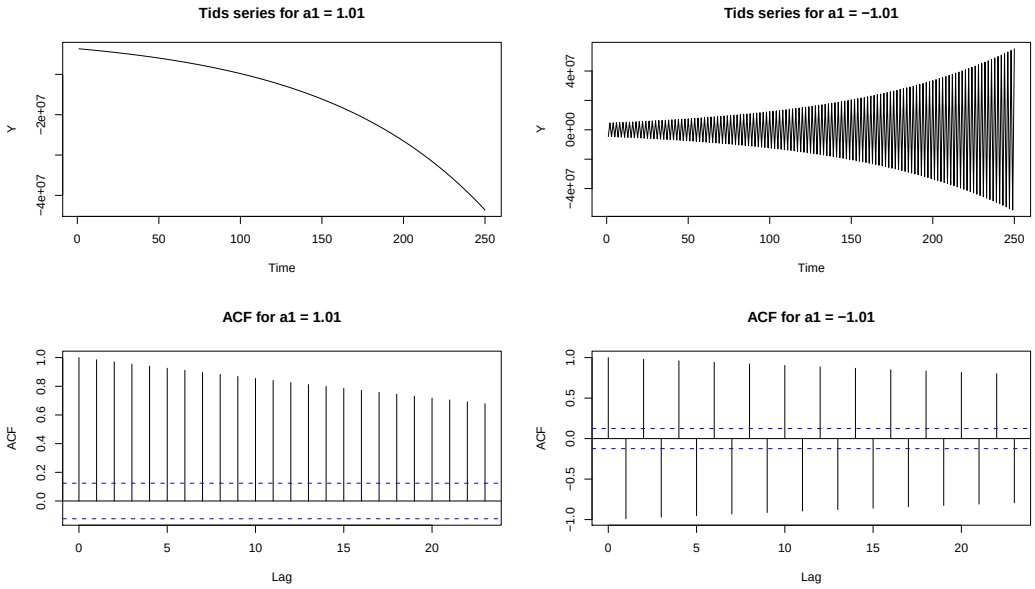

Figure 5: *Simulation of the AR(1)-process (with* $a_1 = 1.01$ *and* $a_1 = -1.01$*) using the* `filter` *function*

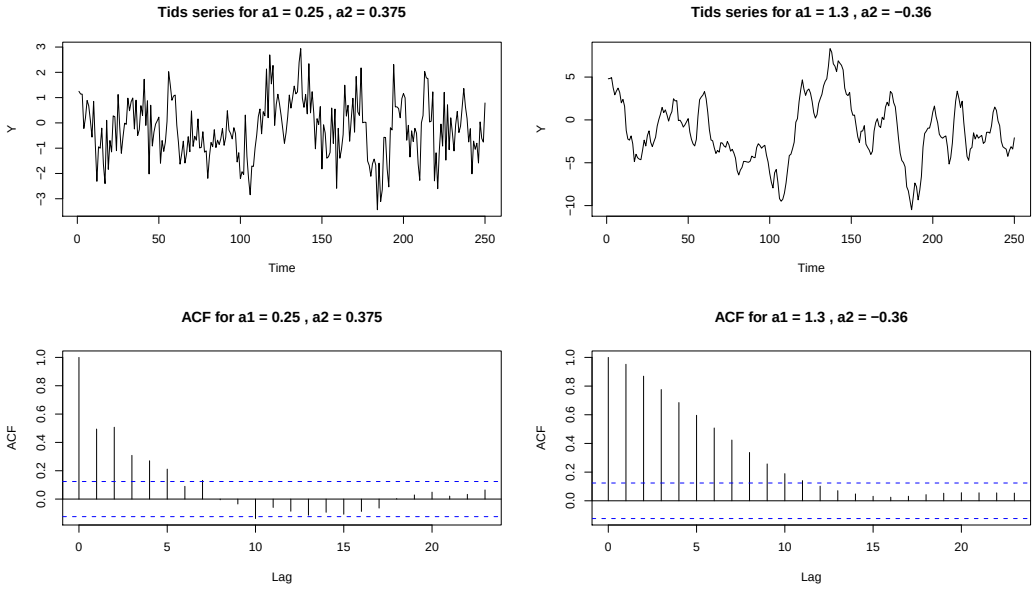

Figure 6: *Simulation of AR(2)-process using the* `filter` *function*

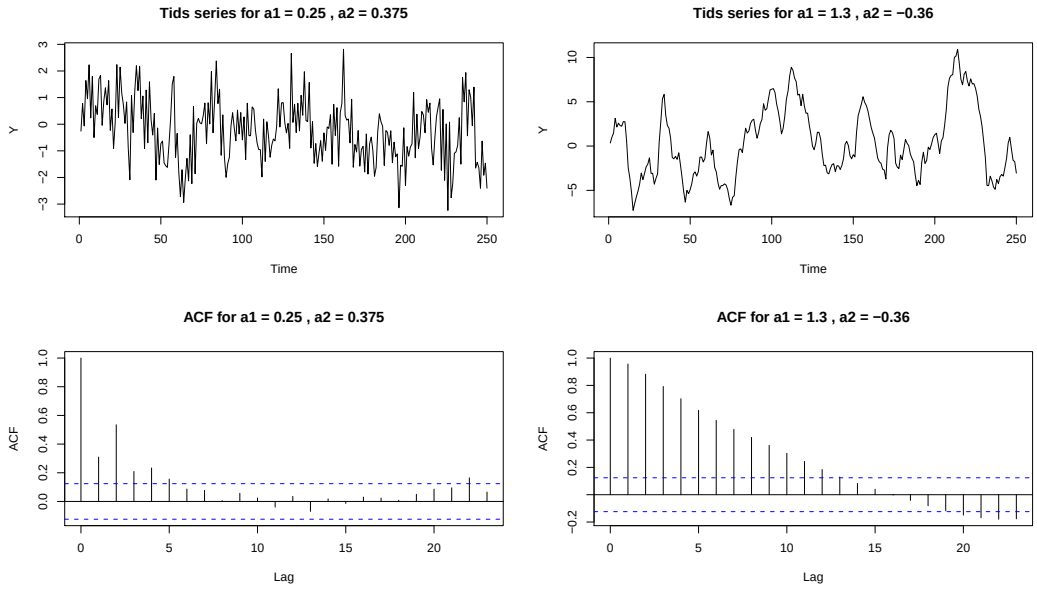

Figure 7: *Simulation of AR(2)-process using the* `arima.sim` *function*

Part 3: Random Walk

We start out by looking at the process given by

$$Y_t = \sum_{s=0}^{t} e_s \tag{2}$$

where e_s is white noise with variance σ_e^2. Then the expectation of the process is

$$\begin{aligned} \mathrm{E}[Y_t] &= \sum_{s=0}^{t} \mathrm{E}[e_s] \\ &= 0 \end{aligned}$$

Since the different e_s are uncorrelated the covariance function can be found as

$$\begin{aligned} \gamma(t,u) &= \mathrm{Cov}[Y_t, Y_u] \\ &= \mathrm{Cov}[\sum_{s=0}^{t} e_s, \sum_{s=0}^{u} e_s] \\ &= \sum_{s=0}^{\min(t,u)} \mathrm{Cov}[e_s, e_s] \\ &= (\min(t,u)+1) \cdot \sigma_e^2 \end{aligned}$$

and then the variance is given by

$$\begin{aligned}\text{Var}[Y_t] &= \gamma(t,t) \\ &= (t+1)\sigma_e^2\end{aligned}$$

Since the covariance depends on the time t we get from definition 5.3 in [1] that the process (2) isn't stationary. To get a better view of the process we now simulate and plot the process. The process can easily be simulated by using the built-in `cumsum` function in R

```
random.walk = function(n=3000, sd=1) {
    ts(cumsum(rnorm(n, mean=0, sd=sd)))
}
```

Using this function we now simulate the process and plot it along with the estimated autocorrelation function. The plot is seen in figure 8 and it is seen that the ACF is almost constant 1 for lags up to 35. This corresponds with the previous paragraph where the process was found to be non-stationary. To further illustrate the point, 6 simulations of the random walk are plotted in figure 9. From the figure it is seen that all 6 processes show some kind of trend, which further illustrates the non-stationarity of the random walk process.

We can now try to remove the non-stationarity by defining a new process as

$$X_t = Y_t - Y_{t-1}$$

and then plot this process along with its estimated auto correlation function. The plot is shown in figure 10 and it is seen that the new process resembles white noise. This isn't surprising since

$$\begin{aligned}Y_t - Y_{t-1} &= \sum_{s=0}^{t} e_s - \sum_{s=0}^{t-1} e_s \\ &= e_t\end{aligned}$$

which shows that the process $\{X_t\}$ is in fact white noise.

Part 4: Seasonal processes

In this section we will simulate multiplicative $(p,d,q) \times (P,D,Q)_s$ seasonal models on the form

$$\varphi(B)\phi(B^s)\nabla^d\nabla_s^D Y_t = \theta(B)\Theta(B^s)\varepsilon_t$$

Since the `arima.sim` function can only simulate non-seasonal models, we have to expand the operator polynomials for some of the models before simulating them in R. The 6 models we are going to simulate are

- **The** $(1,0,0) \times (0,0,0)_{12}$ **model** with $\varphi_1 = 0.9$
  This is just a regular AR(1) process so no expansion is needed

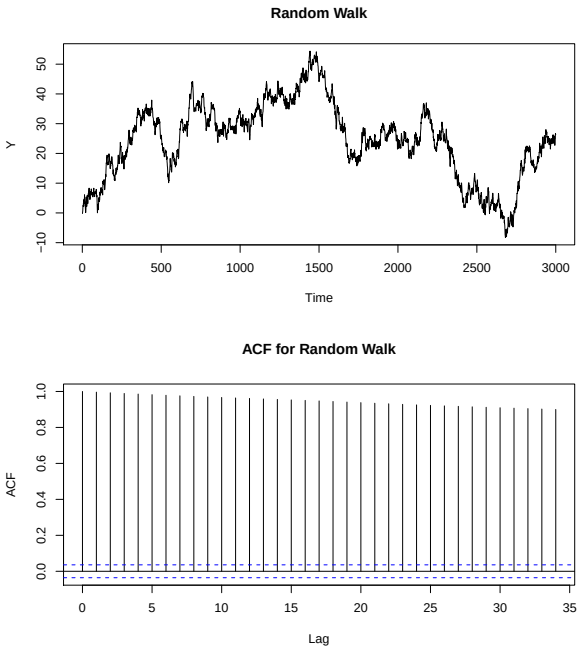

Figure 8: *Simulation of the Random Walk process using the* `rnorm` *and* `cumsum` *function in R*

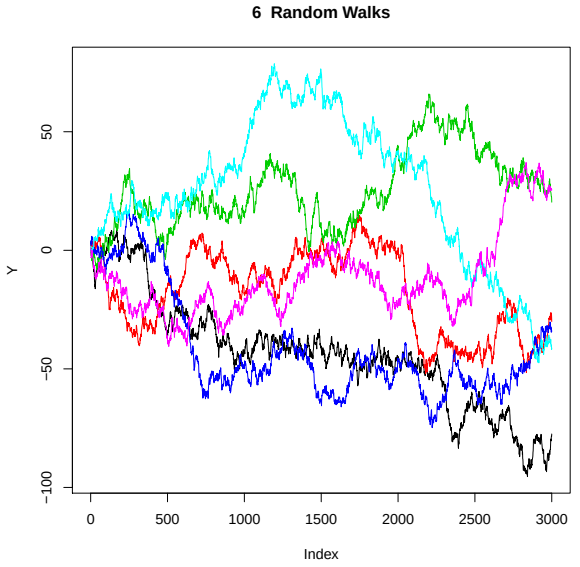

Figure 9: *Simulation of 6 Random Walk processes*

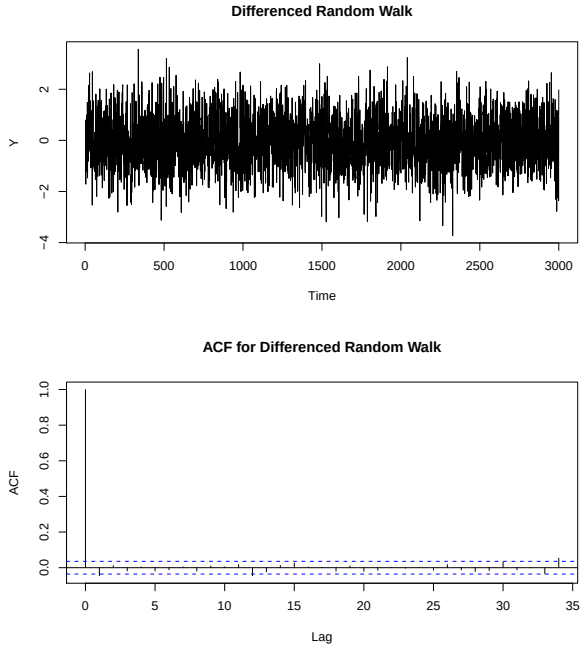

Figure 10: *Simulation of the process constructed by taking first differences of the Random Walk process*

- **The** $(0,0,0) \times (1,0,0)_{12}$ **model** with $\phi_1 = 0.7$
  This is a seasonal AR(1) process but can also be seen as an AR(12) process, so no expansion is needed

- **The** $(1,0,0) \times (0,0,1)_{12}$ **model** with $\varphi_1 = 0.9, \Theta_1 = 0.4$
  This can be regarded as an ARMA(1,12) process and no expansion is needed

- **The** $(1,0,0) \times (1,0,0)_{12}$ **model** with $\varphi_1 = 0.9, \phi_1 = 0.7$
  This model needs to be expanded to give

$$\begin{aligned}(1+\varphi_1 B)(1+\phi_1 B^{12})Y_t = \varepsilon_t \quad &\Leftrightarrow \\ (1+\varphi_1 B+\phi_1 B^{12}+\varphi_1\phi_1 B^{13})Y_t = \varepsilon_t\end{aligned}$$

  which is an AR(13) process

- **The** $(0,0,1) \times (0,0,1)$ **model** with $\theta_1 = 0.4, \Theta_1 = 0.3$
  This model should also be expanded to give

$$\begin{aligned}Y_t = (1+\theta_1 B)(1+\Theta_1 B^{12})\varepsilon_t \quad &\Leftrightarrow \\ Y_t = (1+\theta_1 B+\Theta_1 B^{12}+\theta_1\Theta_1 B^{13})\varepsilon_t\end{aligned}$$

  which is an MA(13) process

- **The** $(0,0,1)\times(1,0,0)_{12}$ **model** with $\theta_1 = 0.4, \phi_1 = 0.7$
  This model can be seen as an ARMA(12,1) process and needs no expansion

With the expansions above, the 6 models are now simulated and the results are plotted in figure 11 and figure 12. From the figures a few things are worth noticing.

- **The** $(1,0,0)\times(0,0,0)_{12}$ **model** with $\varphi_1 = 0.9$ is just an AR(1)-process and therefore the ACF decays. But the decay is slow corresponding to a root with absolute value close to 1. This is the same result as was found in part 2.

- **For the** $(0,0,0)\times(1,0,0)_{12}$ **model** with $\phi_1 = 0.7$ we see a high value of the ACF for lag 12, which we would expect since the seasonal length is 12. The characteristic equation for the model have 12 complex roots r with $|r| = 0.971$ for all roots. The process is stationary and what is seen in the ACF could be a slow decaying sine function.

- **The** $(1,0,0)\times(0,0,1)_{12}$ **model** with $\varphi_1 = 0.9, \Theta_1 = 0.4$ is an ARMA(1,12) and as mentioned on page 127 in [1] the ACF consists of damped exponential and harmonic functions only from lag $k = q + 1 - p$, which in this case is lag 12. This may explain the slow decay for this stationary process.

- **The** $(1,0,0)\times(1,0,0)_{12}$ **model** with $\varphi_1 = 0.9, \phi_1 = 0.7$ is an AR(13) process and by solving
  $$z^{13} + 0.9z^{12} + 0.7z + 0.63 = 0$$
  it is found that the characteristic equation has one negative real root and 6 pairs of complex conjugated roots. By the comments on page 122 in [1] we would expect a combination a shifting positive and negative exponentially decreasing function and damped waves. This is excatly what is seen in the plot.

- **The** $(0,0,1)\times(0,0,1)$ **model** with $\theta_1 = 0.4, \Theta_1 = 0.3$ is a MA(13) process and from table 6.1 in [1] we should expect the ACF to be 0 for lags greater than 13. Except from lag 15 this seems to be the case.

- **The** $(0,0,1)\times(1,0,0)_{12}$ **model** with $\theta_1 = 0.4, \phi_1 = 0.7$ is an ARMA(12,1) process and from page 126 in [1] we should expect that it consists of damped exponential and sine functions. In this case it seems to consist of sine functions.

To conclude it is seen that the ACF for lag 12 is high in model 2, 5 and 6. Model 1 isn't a seasonal model so it is natural that the ACF isn't high in this model at lag 12. A slightly higher correlation at lag 12 is also seen in model 4, although it isn't significant. Also the rate of decay in model 3 seems to slow down at lag 12, but this is also not significant.

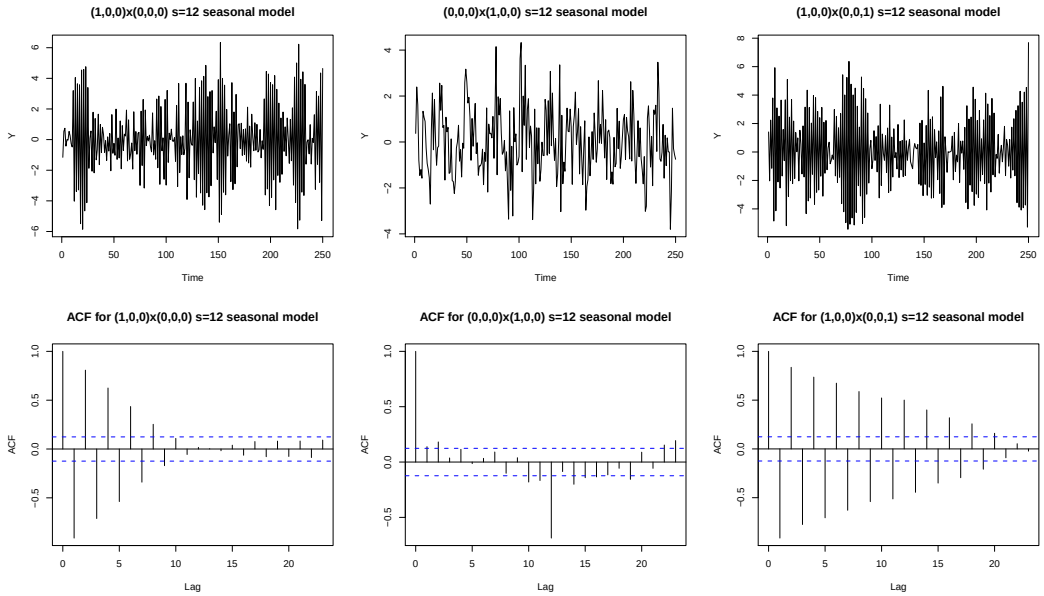

Figure 11: *Simulation of the first 3 seasonal models in part 4.*

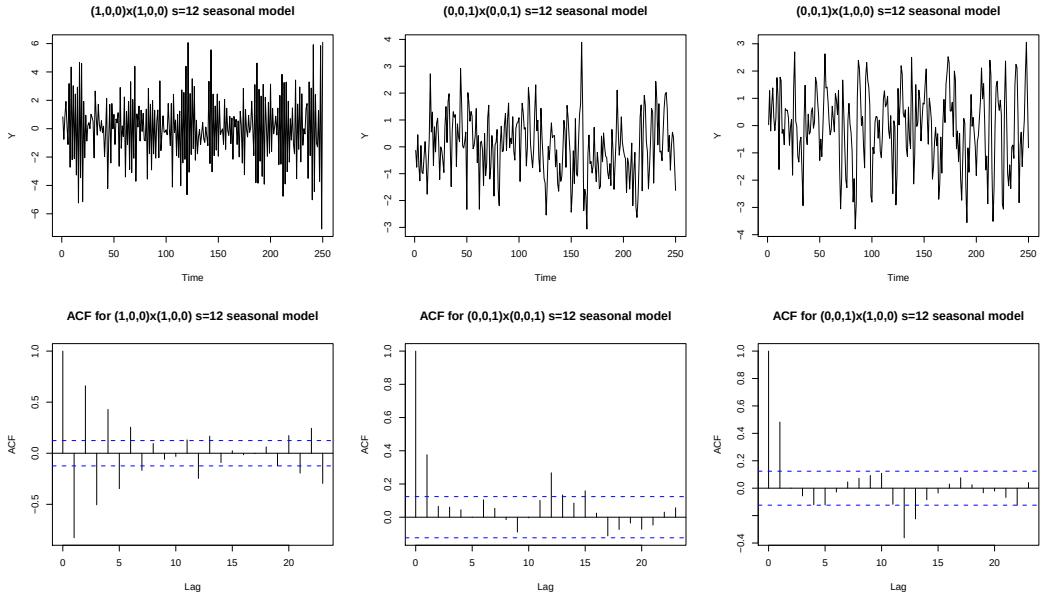

Figure 12: *Simulation of the last 3 seasonal models in part 4.*

Appendices

All R code used for this assignment is included here. All source code incl. latex code for this report can be found at `https://github.com/alphabits/dtu-fall-2011/tree/master/02417/assignment-2`

(part1.R)

```
phi.1 = seq(-2, 2, length.out=200)

const = rep(1, times=200)
quad = phi.1^2/4
linear = abs(phi.1) - 1

plot.stationary = function() {
    plot(phi.1, const, xlim=c(-3,3), ylim=c(-1.5, 1.5), type="l", lty=2,
         xlab="Phi 1", ylab="Phi 2", main="Area with stationary AR(2) process")
    polygon(c(phi.1, rev(phi.1)), c(linear, const), col="gray")
    lines(phi.1, linear, lty=2)
}

plot.damping = function() {
    plot(phi.1, const, xlim=c(-3,3), ylim=c(-1.5, 1.5), xlab="Phi 1",
         ylab="Phi 2", type="l", lty=2, main="Area with damped oscillations for ACF")
    polygon(c(phi.1, rev(phi.1)), c(quad, const), col="gray")
    lines(phi.1, quad, lty=2)
    lines(phi.1, linear, lty=2)
}

save.plot = function(plotfns, filename) {
    pdf(filename)
    plotfns()
    dev.off()
}
```

(ex2.R)

```
y = c(-3, 0, 4, 4, 1, -3, -2, -1, 1, 2)
w = c(1.8, -0.8, 0, 0, 0, 0.2, -0.36, 0.16, 0, 0)
sigma.hat = 0.31

for (k in 1:2) {
    y = c(sum(w*y), y)
    w = c(w, 0)
}

error.variance = (1+w[1]^2)*sigma.hat
conf.step = qnorm(0.975)*sqrt(error.variance)
```

(part2.R)

```
numpoints = 1500
e = rnorm(numpoints)
ind.to.include = (numpoints-249):numpoints


##############################################
# Make common interface for all simulations #
##############################################

my.filter = function (coeffs) {
    ts(filter(e, filter=coeffs, method="recursive")[ind.to.include])
}

my.arima.sim = function (coeffs) {
    arima.sim(model=list(ar=coeffs), n=250)
}

sim = function (simfns, coeffs) {
    ys = list()
    acfs = list()
    i = 1
    for (cs in coeffs) {
        ys[[i]] = simfns(cs)
        acfs[[i]] = acf(ys[[i]], plot=FALSE)
        i = i + 1
    }
    return(list(ys=ys, acfs=acfs, coeffs=coeffs))
}


###################
# AR(1) processes #
###################

coeffs = list(0.1, -0.1, 0.9, -0.9, 1.01, -1.01)
ar1.res = sim(my.filter, coeffs)


###################
# AR(2) processes #
###################

# Roots: (-1/2, 3/4) and (2/5, 9/10)
coeffs.ar2 = list(c(1/4, 3/8), c(13/10, -9/25))
ar2.res = sim(my.filter, coeffs.ar2)


########################
# AR(2) with arima.sim #
########################

# Roots: (-1/20, 1/10) and (5/6, 5/6)
coeffs.ar2.sim = list(c(1/20, 1/200), c(5/3, 25/36))
ar2.sim.res = sim(my.arima.sim, coeffs.ar2)


#################################################
# Helper functions for creating and saving plots #
#################################################
```

```
ar1.filename = "ar1-filter-%s.pdf"
ar2.filename = "ar2-filter-%s.pdf"
ar2.sim.filename = "ar2-sim-%s.pdf"

plot.timeseries = function (series, acfs, coeffs) {
    cols = length(acfs)
    par(mfcol=c(2,cols))
    for (ci in 1:cols) {
        cs = coeffs[[ci]]
        coeff.text = paste("for a1 =", cs[1])
        if (length(cs) == 2) {
            coeff.text = paste(coeff.text, ", a2 =", cs[2])
        }
        plot(series[[ci]], ylab="Y",
             main=paste("Time series", coeff.text))
        plot(acfs[[ci]], main=paste("ACF", coeff.text))
    }
}

save.plots = function (sim.res, filename) {
    numplots = length(sim.res$ys)
    for (i in 1:ceiling(numplots/2)) {
        s = 2*i - 1
        e = if (s == numplots) s else s+1
        ind = s:e
        pdf(sprintf(filename, i), 12, 7)
        with(sim.res, {
            plot.timeseries(ys[ind], acfs[ind], coeffs[ind])
        })
        dev.off()
    }
}
```

(part3.R)

```
random.walk = function(n=3000, sd=1) {
    ts(cumsum(rnorm(n, mean=0, sd=sd)))
}

series = lapply(rep(3000, 6), random.walk)
diffed.series = lapply(series, diff)

minmaxfns = function(f, lst) {f(unlist(lapply(lst, f)))}

# Finding the minimum and maximum across
# all time series in series.list
get.limits = function(series.list) {
    lapply(list(min, max), minmaxfns, series.list)
}

plotfns.factory = function(series.list) {
    function(x, y, main, col=1) {
        plot(x, y, main=main, col=col, ylim=unlist(get.limits(series.list)),
             type="l", xlab="Index", ylab="Y")
    };
}

plot.timeseries = function(series.list, main, ind=1:3000) {
```

```
    num.series = length(series.list)
    for (i in 1:num.series) {
        fns = if (i == 1) plotfns.factory(series.list) else lines
        fns(ind, series.list[[i]][ind],
            main=paste(num.series, main), col=i)
    }
}

save.multiple.timeseries = function(series.list, main, filename) {
    pdf(filename)
    plot.timeseries(series.list, main)
    dev.off()
}

save.multiple.random.walks = function() {
    save.multiple.timeseries(series, " Random Walks", "multiple-random-walk.pdf")
}

save.multiple.diffed.random.walks = function() {
    save.multiple.timeseries(diffed.series, " Diffed Random Walks",
                             "multiple-diffed-random-walk.pdf")
}

plot.single = function(ts, filename, main.ts, main.acf) {
    pdf(filename, 8, 9)
    par(mfrow=c(2,1))
    plot(ts, ylab="Y", main=main.ts)
    acf(ts, main=main.acf)
    dev.off()
}

plot.single.random.walk = function() {
    plot.single(series[[1]], "random-walk.pdf", "Random Walk",
                "ACF for Random Walk")
}

plot.single.random.walk.diffed = function() {
    plot.single(diffed.series[[1]], "random-walk-diffed.pdf",
                "Differenced Random Walk", "ACF for Differenced Random Walk")
}
```

(part4.R)

```
zs = function (num) {
    return(rep(0, times=num))
}

model = function(a1, b1) {return(c(a1, zs(10), b1, a1*b1))}

models = list(
    list(ar=-c(0.9)),
    list(ar=-c(zs(11), 0.7)),
    list(ar=-c(0.9), ma=c(zs(11), 0.4)),
    list(ar=-model(0.9, 0.7)),
    list(ma=model(0.4, 0.3)),
    list(ar=-c(zs(11), 0.4), ma=c(0.7))
);

model.descriptions = list(
```

```
    "(1,0,0)x(0,0,0) s=12",
    "(0,0,0)x(1,0,0) s=12",
    "(1,0,0)x(0,0,1) s=12",
    "(1,0,0)x(1,0,0) s=12",
    "(0,0,1)x(0,0,1) s=12",
    "(0,0,1)x(1,0,0) s=12"
)

plot.models = function(ind) {
    par(mfcol=c(2,3))
    for (i in ind) {
        series = arima.sim(250, model=models[[i]])
        main = paste(model.descriptions[i], "seasonal model")
        plot(series, main=main, ylab="Y")
        acf(series, main=paste("ACF for", main))
    }
}

save.plots = function(ind) {
    num = min(ind[1], 2)
    filename = sprintf("seasonalmodels-%s.pdf", num)
    pdf(filename, 12, 7)
    plot.models(ind)
    dev.off()
}
```

References

[1] Henrik Madsen, *Time Series Analysis*. Chapman & Hall/CRC, 1st Edition, 2008.