Problem Set 1

Lorenzo Orsenigo, Luca Geminiani

October 17^{th}, 2023

Exercise 1

We want to generate 500 observations from an AR(1): $Y_t = 0.4Y_{t-1} + \varepsilon_t$

```
AR1 = [1 -0.4];
```

In which $E(Y_t) = 0$, $\sigma_\varepsilon^2 = 0.2$, and we are assuming that the forcing term is normally distributed.
To check for stationarity we look at the roots of the characteristic equation $1 - 0.4z = 0$

```
rt = roots(fliplr(AR1))
```

As the result is $2.5 > 1$, the process is stationary.
To confirm the result we check that the original polynomial is obtained when using the command `poly`

```
poly(1./rt)
```

which indeed returns the original polynomial.
We could also have equivalently checked stationarity by looking at the root of $1x - 0.4 = 0$ (stationary if the root is smaller than 1).

First method

We can generate the observations by generating a Gaussian white noise random draw from a Normal (0,0.2) and then using a `for` loop

```
ar = zeros(500,1);
mu = 0;          % Mean
sigma2 = 0.2;      % Variance
epsilon = mu + sqrt(sigma2) * randn(500,1);
ar(1)=epsilon(1);

for t = 2:500
    ar(t) = 0.4*ar(t-1) +epsilon(t);
end
```

More concisely we could also have used (not in Matlab script)

```
ar = zeros(500,1);
ar(1) = normrnd(0,0.2)
for t = 2:500
    ar(t) = 0.4*ar(t-1) + normrnd(0,0.2);
end
```

We can then look at the graph of the sample autocorrelation function (ACF) generated with.

```
figure; autocorr(ar,[],1)
```

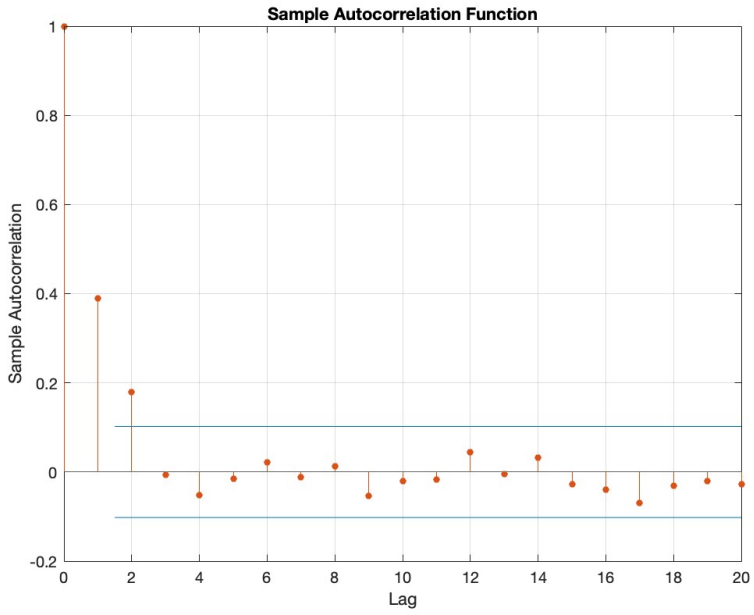


Figure 1: ACF of the generated AR(1) with a 95% confidence interval in light-blue

The ACF geometrically decays and is not truncated, as one should expect.

Second method

In order to generate the observations we can also use the command `filter`.
We exploit the previous simulation of a Gaussian(0,0.2) white noise process, `epsilon`, then use filter to generate AR(1).

```
y1 = filter(1,AR1,epsilon)
```

We can hence inspect the ACF process via the same command as before

```
figure; autocorr(y1,[],1)
```

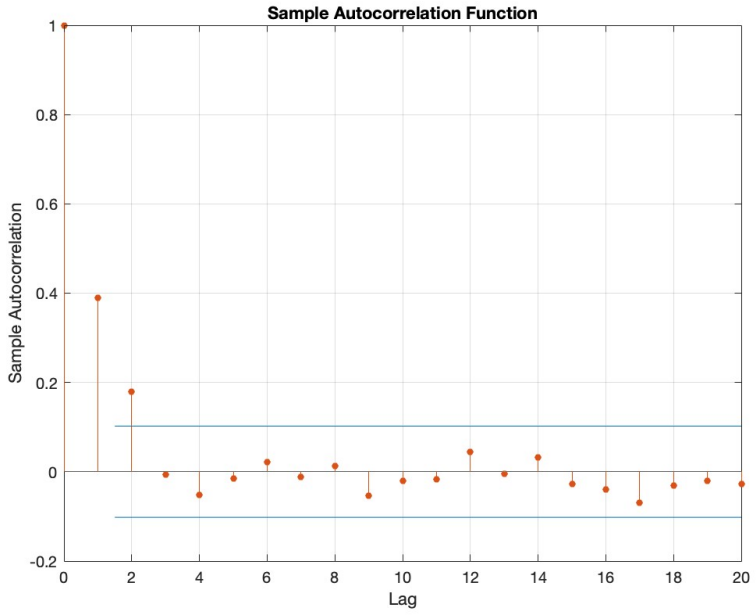


Figure 2: ACF of the generated AR(1) with a 95% confidence interval in light-blue

and we can also observe analogously a geometrically decaying path.

Comparison

As it was already evident from Figure 1 and Figure 2. the two generated processes appear to be identical. We can more rigorously verify this by using

```
y1 == ar
```

which returns the logical operator ”1”, or true. Hence the two outputs, utilizing the same forcing variables, are identical.

We can check this in more detail via

```
if y1 == ar
    disp('Identical Vectors')
else
    disp('Some elements differ')
end
```

in which if only one element among all the 500 is different, the logical check will fail and ’Some elements differ’ will be printed.

These commands indeed return ’Identical Vectors’: the two methods create identical processes.

Exercise 2

In order to generate data from an AR(1) with $\phi = 0.6$, $\sigma^2 = 0.4$ and $E(Y_t) = 3$, as well as a starting condition of 10, we can use the `for` in the following manner:

```
ar = zeros(500,1);
mu = 3;          % Mean
sigma2 = 0.4;      % Variance
epsilon= mu + sqrt(sigma2) * randn(500,1);
ar(1)=10;

for t = 2:500
    ar(t) = 0.6*ar(t-1)+epsilon(t);
end
```

If the starting condition is far from the unconditional mean, the process is initialized at a point that deviates significantly from the expected behaviour of its values. However, the initial imposition is innocuous as long as $|\phi| < 1$. If instead $|\phi| \geq 1$ or $|\phi| \approx 1$, then the initial condition impacts the process and hence all the following results. An immediate solution in this case is to repeat the generation of the data by making the first element of the sample be drawn, as well, from a Normal(3,0.4). Alternatively, we could impose the first value of the sample y as equal to $E(y_t)$.
This same process with the same initial imposition could also be generated with the command `filter`:

```
AR1 = [1 -0.6];
epsilon(1)=10;
y1= filter(1,AR1,epsilon);
```

The considerations are identical, in this case to correct the deviating initial imposition we could simply omit the second line of code `epsilon(1)=10;` or set the first element of the vector epsilon equal to $E(y_t)$.

Exercise 3

We want to generate 500 observations from a MA(1): $x_t = 0.3\varepsilon_{t-1} + \varepsilon_t$

```
MA1 = [1 0.3];
```

In which the variance of the white noise is $\sigma_\varepsilon^2 = 0.3$.

First method

To build the observations we generate a Gaussian white noise random draw from a Normal (0,0.3) and then use a `for` loop

```
ma = zeros(500,1);
mu = 0;          % Mean
sigma2 = 0.3;      % Variance
epsilon = mu + sqrt(sigma2) * randn(500,1);
ma(1)=epsilon(1);

for t = 2:500
    ma(t) = 0.3*epsilon(t-1) + epsilon(t);
end
figure; autocorr(ma,[],2)
```

We can then look at the graph sample autocorrelation function (ACF) generated with

```
figure; autocorr(ma,[],1)
```

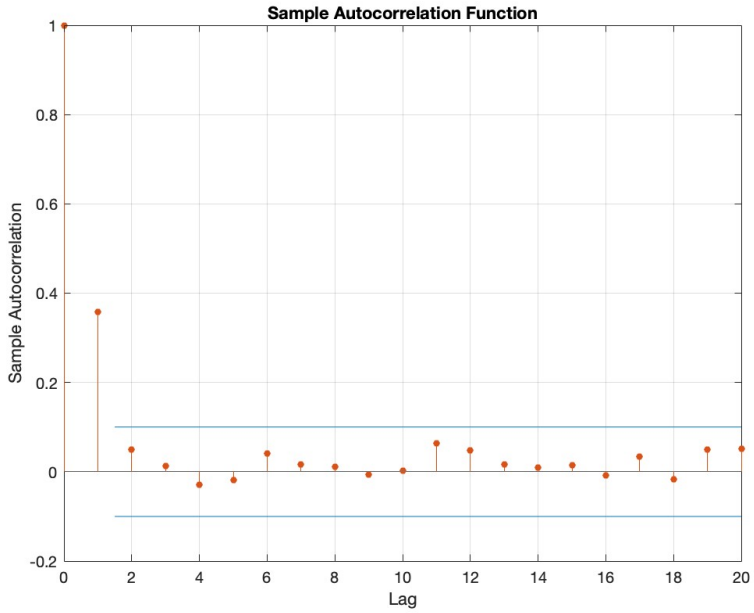


Figure 3: ACF of the generated MA(1) with a 95% confidence interval in light-blue

which shows how it is truncated after 1 step ahead (MA(1)), instead of geometrically decaying as in AR processes.

Second method

In order to generate the 500 observations we may also use the command `filter`.
Again we use the random draw generated from a Normal (0,0.3), `epsilon`, and then employ the command to apply the MA(1) process to the generated white noise

```
x1 = filter(MA1,1,epsilon);
```

We can hence inspect ACF of the process via the same command as before

```
figure; autocorr(x1,[],1)
```

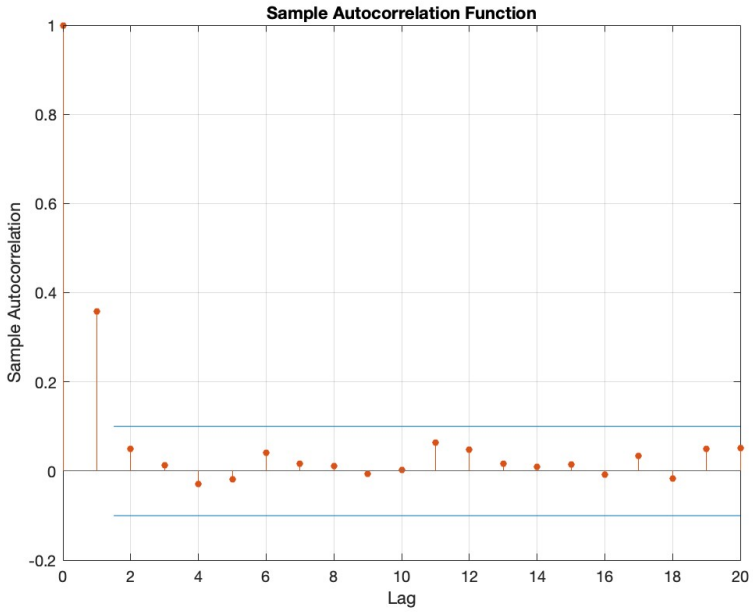


Figure 4: ACF of the generated MA(1) with a 95% confidence interval in light-blue

and we can observe the same path, analogously truncated.

Exercise 4

We want to write a function that, when you input

- number of observations, `T`
- variance of whine noise forcing variable `sigma`
- a vector of AR coefficients `phi` $= [\phi_1\ \phi_2\ ...\ \phi_p]$
- a vector of MA coefficients `theta` $= [\theta_1\ \theta_2\ ...\ \theta_q]$

provides as output

- the realizations of the $ARMA(p,q)$, a vector `x`

- the realization of the white noise, a vector `epsilon`

The function is the following:

```
function[x,epsilon] = for_arma(T,sigma,phi,theta)

epsilon=sqrt(sigma) * randn(T, 1);
x=zeros(T,1);
x(1)=epsilon(1);
Q=length(theta);
P=length(phi);
A=zeros(P,1);
M=zeros(Q,1);

for t=2:T
    for p=1:P
        if t-1>=p
            A(p)=phi(p)*x(t-p);
        else
            A(p)=0;
        end
    end
    for q=1:Q
        if t-1>=q
            M(q)=theta(q)*epsilon(t-q);
        else
            M(q)=0;
        end
    end
    x(t)=sum(A)+sum(M)+epsilon(t);
end
end
```

To verify its well-functioning, we run two tests. First we generate an ARMA(2,2) and look at the ACF:

```
phi=[0.3 0.4];
theta=[0.3 0.4];
[x,epsilon] = for_arma(500,0.2,phi,theta);
figure;autocorr(x,[],2)
```

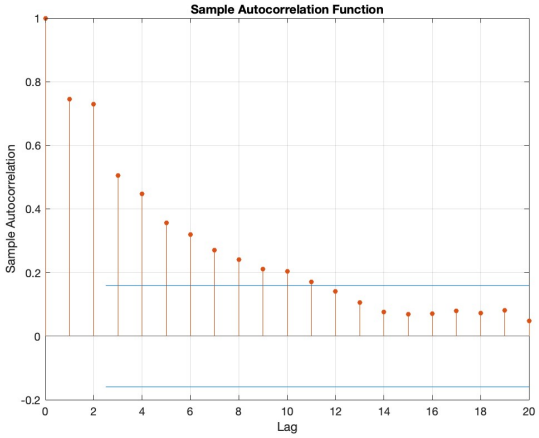


Figure 5: ACF of the generated ARMA(2,2) with a 95% confidence interval in light-blue

From Figure 5 we can clearly see that, due to the AR part of the ARMA process, the ACF is not truncated after the second lag. Second, by using the same function and eliminating the AR part of the process, hence by adding only the (three in this example) MA coefficients, we generate an MA(3) process, whose ACF is indeed truncated after the third lag:

```
phi=[];
theta=[0.3 0.4 0.2];
[x,epsilon] = for_arma(500,0.2,phi,theta);
figure;autocorr(x,[],3)
```

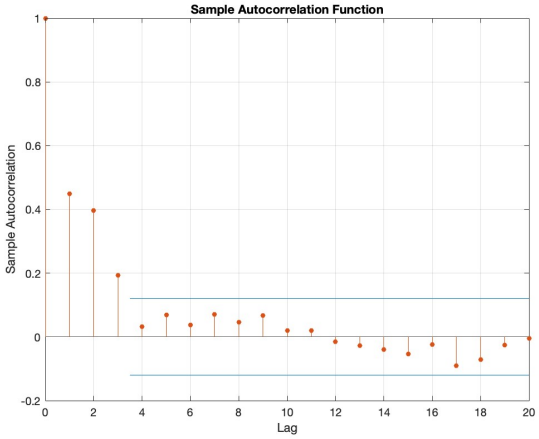


Figure 6: ACF of the generated MA(3) with a 95% confidence interval in light-blue

Exercise 5

a) First, we compute the empirical distribution of the OLS estimator of an AR(1) process, $x_t = \phi x_{t-1} + \varepsilon_t$, with $\phi = 0.4$, $\sigma_\varepsilon^2 = 0.4$ and $T = 250$.
As a primary step, we build a function "OLS" for running the regression. If we input realisations on dependent and independent variables, we obtain the estimate of the coefficients of the explanatory variables, the variance of the coefficients and the residuals of the model.

```
function [b,res,cov_b] = OLS(y,x)

b = inv(x'*x)*x'*y ;
res = y-x*b;
cov_b = inv(x'*x)*cov(res);
end
```

Then we can use it to find the empirical distribution by running 10000 simulations of the estimation of the coefficient ϕ

```
emp_ols = zeros(10000,1); %vector of coefficients
for i = 1:10000
    ar = zeros(250,1);
    ar(1) = normrnd(0,0.4);
    epsilon = sqrt(0.4) * randn(250, 1);
    for t = 2:250
        ar(t) = 0.4*ar(t-1) + epsilon(t);
    end
    arx = ar(1:249);
    ary = ar(2:250);
    [b,res,cov_b] = OLS(ary,arx);
    emp_ols(i) = b;
end
```

We can visualise the resulting empirical distribution with the following command:

```
figure; histogram(emp_ols)
```

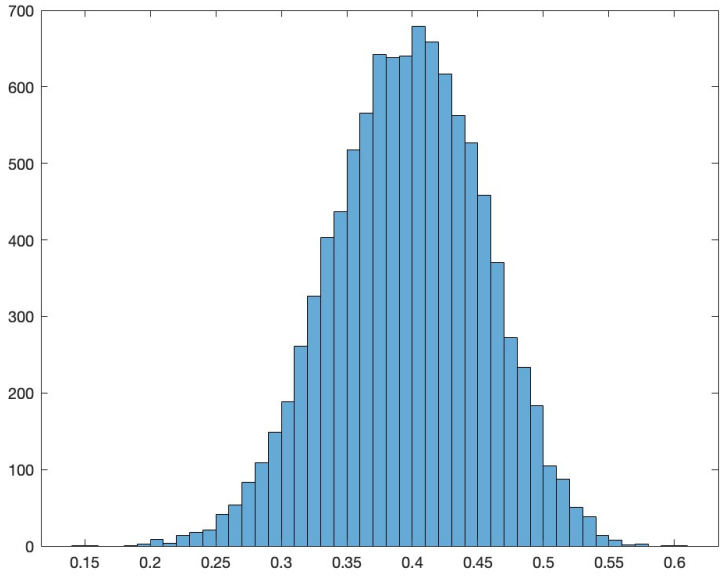


Figure 7: empirical distribution of the OLS coefficient

b) Then, we decrease the number of observations to T=200 and run the simulations to obtain the empirical distribution. This time we also build a t-test on the ϕ parameter to see how many times we reject the null hypothesis $H_0: \phi = 0$ against the alternative $H_1: \phi \neq 0$ at a 95% confidence level

```
emp_ols = zeros(10000,1); %vector of coefficients
t_test = zeros(10000,1); %vector of t-test coefficients
for g = 1:10000
    ar = zeros(200,1);
    ar(1) = normrnd(0,0.2);
    epsilon = sqrt(0.2) * randn(200, 1);
    for t = 2:200
        ar(t) = 0.4*ar(t-1) + epsilon(t);
    end
    arx = ar(1:199); %independent vars
    ary = ar(2:200); %dependent vars
    [b,res,cov_b] = OLS(ary,arx);
    emp_ols(g) = b;
    t_test(g) = b/sqrt(cov_b) ;
end

critical = tinv(1-(0.05/2),200-2);
sum(abs(t_test)>critical)/10000
```

The result we obtain is 0.9998, which tells us that we rightly reject the null hypothesis 99.98% of the times,

we hence have a 0.02% probability of committing a type II error. This makes sense because the 'true' model has a ϕ parameter notably different from 0.

Exercise 6

Exploiting the OLS function created in the previous exercise we compute the empirical distribution of the OLS estimator in the case of an AR(1) with $\phi = 0.9$ and $T = \{50, 100, 200, 1000\}$.

```
G=[50,100,200,1000];
emp_ols_results = cell(length(G), 1);

for g= 1:length(G)
    T = G(g);
    emp_ols = zeros(10000,1);
    for i = 1:10000
        ar = zeros(T,1);
        ar(1) = normrnd(0,0.2);
        epsilon = sqrt(0.2) * randn(T, 1);
        for t = 2:T
            ar(t) = 0.9*ar(t-1) + epsilon(t);
        end
        arx = ar(1:T-1); %independent vars
        ary = ar(2:T); %dependent vars
        [b,res] = OLS(ary,arx);
        emp_ols(i) = b ;
    end
    emp_ols_results{g} = emp_ols ;
end
```

Once having obtained the empirical distributions of the OLS estimators we can visualise them to see how they change in the four cases, with different T, via the following script:

```
for g = 1:length(G)
    T = G(g);
    subplot(2, 2, g);
    histogram(emp_ols_results{g}, 30, 'Normalization','probability');
    title(['T = ' num2str(T)]);
end
```

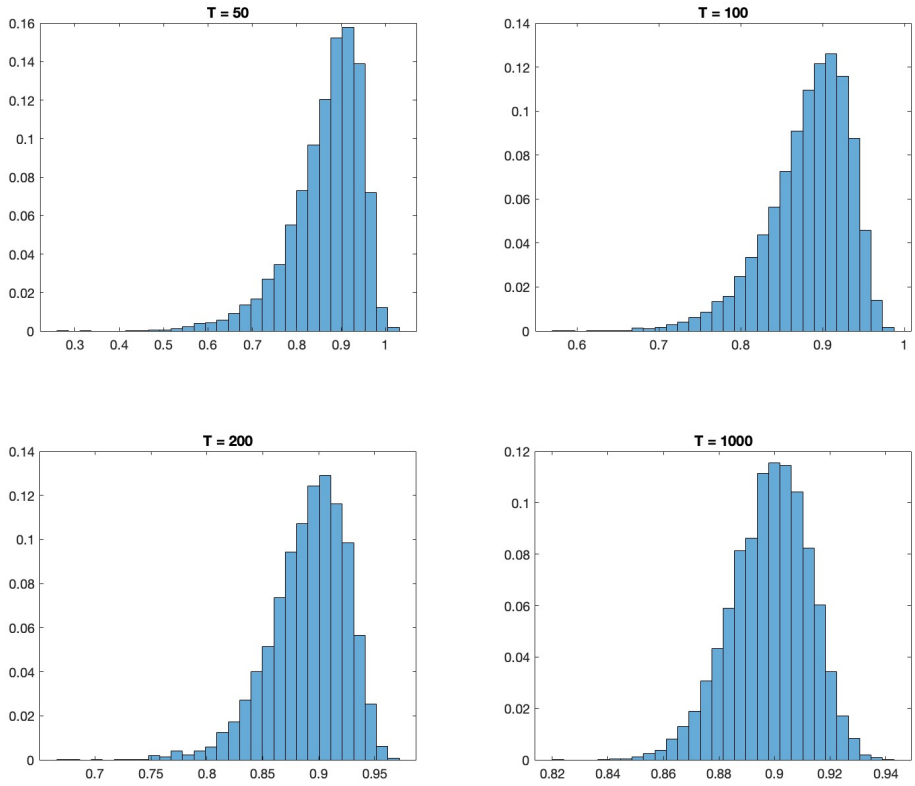


Figure 8: empirical distributions of the OLS estimator with different T

From Figure 8 we observe what is expected when the number of observations is increased: the expected value of the estimator converges to its true value, 0.9, and its variance decreases.

Exercise 7

Consider an MA(1) process generating x_t

$$x_t = \varepsilon_t + \theta \varepsilon_{t-1}$$

with $\theta = 0.6$ and $T = 250$, and assume as variance of the forcing term $\sigma_\varepsilon^2 = 0.3$. We want to find the empirical distribution of the OLS estimator for a in the following regression: $x_t = a x_{t-1} + v_t$

```
a_ols = zeros(1000,1);
for i = 1:1000
    ma = zeros(250,1);
```

```
    mu = 0;          % Mean
    sigma2 = 0.3;      % Variance
    epsilon = mu + sqrt(sigma2) * randn(250,1);
    ma(1)=epsilon(1);
    for t = 2:250
        ma(t) = epsilon(t) + 0.6*epsilon(t-1);
    end
    xt_lagged = ma(1:249); %independent vars
    xt = ma(2:250); %dependent vars
    [b,res] = OLS(xt,xt_lagged);
    a_ols(i) = b ;
end
```

To provide the mean as a summary of the obtained distribution, we use `mean`

```
mean(a_ols)
```

In this case, the result is 0.4407; remarkably different from its theoretical value, 0. By running the script a few other times we get 0.4388, 0.4404, 0.4384, 0.4409. By increasing the number of observations, hence T, for example to T=250000, we can see that the mean converges towards a certain value.

```
a_ols = zeros(1000,1);
for i = 1:1000
    ma = zeros(250000,1);
    mu = 0;          % Mean
    sigma2 = 0.3;      % Variance
    epsilon = mu + sqrt(sigma2) * randn(250000,1);
    ma(1)=epsilon(1);
    for t = 2:250000
        ma(t) = epsilon(t) + 0.6*epsilon(t-1);
    end
    xt_lagged = ma(1:249999); %independent vars
    xt = ma(2:250000); %dependent vars
    [b,res] = OLS(xt,xt_lagged);
    a_ols(i) = b ;
end
mean (a_ols)
```

Whenever we run this script, the result is almost constantly the same: 0.4412. But what does this value represent?

The problem is that the model we are using for the regression is wrong. We are in a case of omitted variables, which indeed leads to biased estimators. The v_t in our regression is indeed equal to what is missing

$$v_t = \theta\varepsilon_{t-1} + \varepsilon_t$$

therefore, since

$$\hat{a} \quad \rightarrow \quad \frac{cov(x_t, x_{t-1})}{var(x_{t-1})}$$

the estimator $\hat{a}$ will not converge to a, whose value is 0, because

$$\frac{cov(x_t, x_{t-1})}{var(x_{t-1})} = a + \theta \frac{cov(x_{t-1}, \varepsilon_{t-1})}{var(x_{t-1})}$$

Hence $\hat{a}$ has an asymptotic bias of

$$\theta \frac{cov(x_{t-1}, \varepsilon_{t-1})}{var(x_{t-1})} = \theta \frac{\sigma_\varepsilon^2}{\theta^2 \sigma_\varepsilon^2 + \sigma_\varepsilon^2}$$

which, if we substitute the values assumed at the beginning, is indeed equal to 0.4412.

Exercise 8

Empirical distribution

a) We exploit the usual method to find the empirical distribution of an AR(1) with $\phi = 1$, $T = 250$ and $\sigma_\varepsilon^2 = 0.2$

```
emp_olsphi = zeros(1000,1);
for i = 1:1000
    ar = zeros(250,1);
    ar(1) = normrnd(0,0.2);
    epsilon = sqrt(0.2) * randn(250, 1);
    for t = 2:250
        ar(t) = ar(t-1) + epsilon(t);
    end
    arx = ar(1:249);
    ary = ar(2:250) ;
    [beta] = OLS(ary,arx);
    emp_olsphi(i) = beta;
end
```

Estimates of ϕ are reasonably close to one as we can see from Figure 9

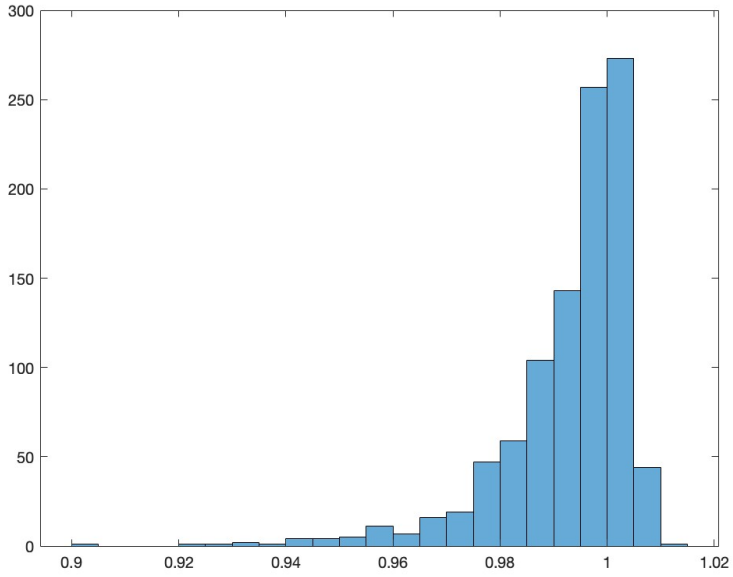


Figure 9: empirical distribution of the OLS estimator

T-test

b) By looking at the following regression: $\Delta y = \alpha + \beta y_{t-1} + \varepsilon_t$ we want to the test the null hypothesis $H_0 : \rho = \phi - 1 = 0$ against the alternative $H_1 : \rho < 0$.

First, we pefrom the t-test on the differenced equation by using the standard normal distribution's critical values.

```
df_test = zeros(10000,1);

for i = 1:10000
    ar = zeros(250,1);
    ar(1) = normrnd(0,0.2);
    epsilon = sqrt(0.2) * randn(250, 1);
    for t = 2:250
        ar(t) = ar(t-1) + epsilon(t);
    end
    arx = [ones(249,1) ar(1:249)];
    ary = ar(2:250) - arx(:,2) ; %differenced dependent variable
    [beta,res,cov_b] = OLS(ary,arx);
    df_test(i) = (beta(2)-0)/sqrt(cov_b(2,2)); %the null is rho = 0
end

sum(df_test<-1.645)/10000
```

In our case we end up rejecting H_0 46.11% of times, we hence reject the null hypothesis too often, since we should have a type I error happening only 5% of times. This shows that the OLS estimator distribution is not asymptotically normal, and by ignoring this fact we would consider the model as stationary more than we should. We can indeed look at the distribution of the t-statistic and see that is not centered in 0, thus behaving differently from a standard normal.

```
hist(df_test)
```

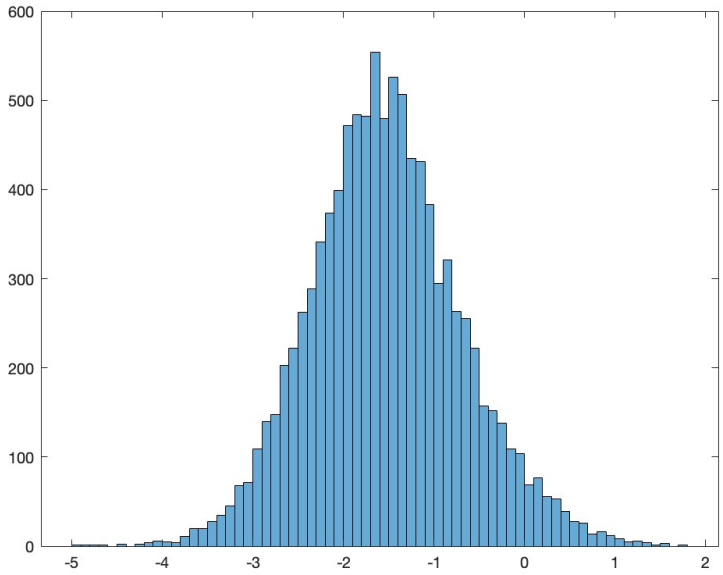


Figure 10: empirical distribution of the t-statistic

If instead we use the Dickey-Fuller distribution (with constant term), we should reject the null hypothesis in a 0.05 level test if the standardised estimate is smaller than 2.88 (the 0.05 percentile in the tabulation of DF in the case of a regression with a constant and T=250). We can immediately verify that, this way, we get approximately the correct type I error percentage.

```
sum(df_test<-2.88)/10000
```

The result in our case is 0.0475, approximately equal to the expected 0.05 we should get by testing correctly our hypotheses. This is sensible as the theoretical model was, in fact, non stationary, being a random walk with drift.

Comparison with DF distribution

We can now more clearly see what we anticipated before by looking at several percentiles of the empirical test statistic under the null hypothesis and comparing them with those of the Dickey-Fuller distribution. With the following command:

```
percentiles = prctile(df_test, [1, 2.5, 5, 10]);
```

we obtain $-3.4606, -3.1214, -2.8650, -2.5649$.
Whilst the values tabulated by DF in the case of the regression including the drift term, with sample size T=250, are: (-3.46, -3.14, -2.88, -2.57); very similar to those of the empirical t-distribution. This confirms that we should not compare the t-statistic to the standard normal distribution but to the Dickey-Fuller distribution including a constant term. The distribution of our t-statistic approximates the DF one including the drift term.

Exercise 9

Monte-Carlo Experiment

Consider a random walk with drift

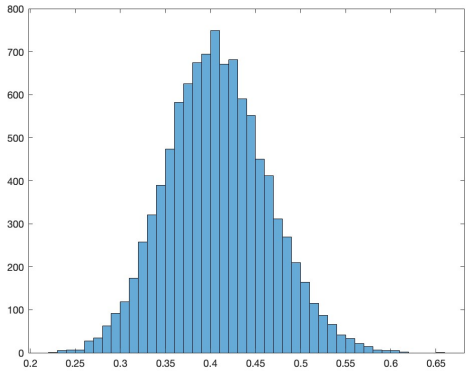
$$x_t = \delta + x_{t-1} + \varepsilon_t$$

with $T = 250$ and $\delta = 0.4$.
In order to study the performance of the Dickey-Fuller test, with $H_0: \phi = 1$ and $H_0: \phi < 1$, we perform a Monte-Carlo experiment.

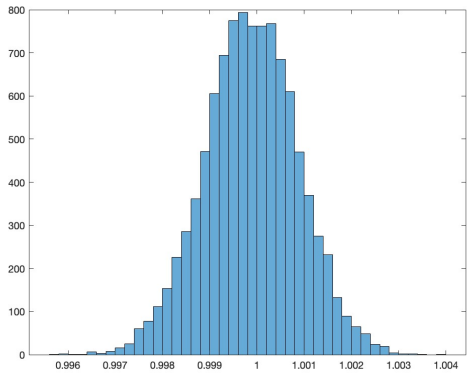
```
T= 250; %number of obs.
delta = 0.4; %drift parameter
for i = 1:10000
    rw = zeros(T,1);
    rw(1) = normrnd(0,0.2);
    epsilon = sqrt(0.2) * randn(T, 1);
    for t = 2:T
        rw(t) = delta + rw(t-1) + epsilon(t);
    end
    rwx = [ones(T-1,1) rw(1:T-1)];
    rwy = rw(2:T);
    [b,res,cov_b] = OLS(rwy,rwx); % b(1) is the est. of the intercept
    B_montecarlo(i,:) = b';
    df_test_1(i) = (b(2)-1)/sqrt(cov_b(2,2)); %the null is phi=1
    df_test_2(i) = (b(1))/sqrt(cov_b(1,1));
end
```

First, we can look at the empirical distribution of the estimators we have just obtained.

```
figure;histogram(B_montecarlo(:,1));
figure;histogram(B_montecarlo(:,2));
```



(a) empirical distribution of $\hat{\delta}$

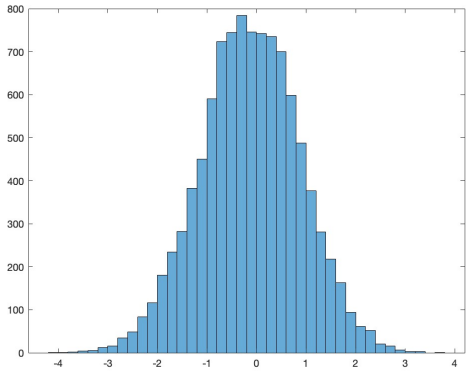


(b) empirical distribution of the $\hat{\phi}$

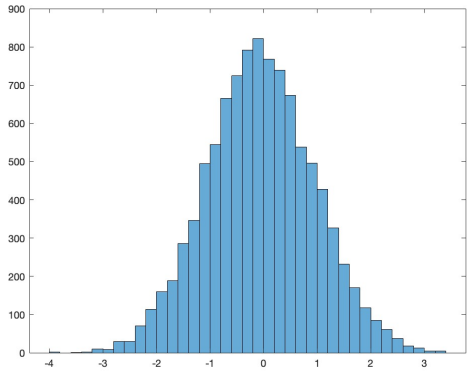
Figure 11: empirical distributions of OLS estimators

Moreover, it is interesting to observe the empirical distribution of the t-statistic as the sample size evolves to T=1000. As the number of observations increases, the distribution gets closer to a standard normal one.

```
figure;histogram(df_test_1);
```



(a) T=250



(b) T=1000

Figure 12: empirical distributions of test statistic

We can see this even more clearly by computing how many times we are rejecting the null hypothesis using the critical values of the standard normal distribution at a 95% level of confidence.

```
sum(df_test_1<-1.645)/10000
```

which, with both sample sizes, is approximately 5%. Importantly, this result is not statistically valid because the t-tests we computed are unilateral and were not performed on a differenced equation. Nonetheless, we have generated the Monte Carlo simulations and assessed how changing the sample size influences

performance of t-tests (and hence df-tests)computed on it. In the next section we will, more rigorously, perform stationarity checks on the model focusing on multiple parameters(including the drift component).

F-test

We want to build an F-test to determine whether in the model H_0 : there is a unit root, or H_1 : there is no unit root, using a χ^2 with one degree of freedom. The choice of degrees of freedom is obtained by differencing the degrees of freedom of the restricted model and of the alternative model.

```
T = 250;
phi = 1;
alpha = 0.5;
sigma = sqrt(0.2);
ols_coeff_rho = zeros(1,1000);
F_stat = zeros(1,1000);
for i = 1:1000
    %AR(1) dataset
    epsilon = sigma*randn(T,1);
    y = zeros(T,1);
    delta_y = zeros(T,1);
    delta_y(1) = y(1);
    for t = 2:T
        y(t) = alpha + phi* y(t-1) + epsilon(t);
        delta_y(t) = y(t) - y(t-1);
    end
    delta_y_t = delta_y(2:end);
    lag = y(1:end-1);
    mdl = fitlm(lag,delta_y_t);
    SSR = mdl.SSE;  % Sum of squared residuals
    SST = mdl.SST;  % Total sum of squares
    df_R = mdl.NumCoefficients - 1;  % Degrees of freedom for the regression
    df_E = mdl.DFE;  % Degrees of freedom for the residuals
    F_stat(i) = ((SST - SSR) / df_R) / (SSR / df_E);
end
```

obtained the test statistics we can now compare them against the critical values to determine how often we reject the the null hypothesis

```
critical = chi2inv(1-0.05, 1);
sum(F_stat>critical)/1000
```

From this result we can see that we correctly commit a type I error only 5% if the times. We graphically show that the distribution of our F-test approximates that of a χ^2 with one degrees of freedom in the following graph:

```
df_chi2 = 1;
figure;
histogram(F_stat, 'Normalization', 'pdf', 'EdgeColor', 'none', 'FaceColor', 'blue',
'DisplayName', 'F-stat Histogram');
hold on;
x = 0:0.01:10;
chi2_pdf = chi2pdf(x, df_chi2);
plot(x, chi2_pdf, 'r-', 'LineWidth', 2, 'DisplayName', 'Chi^2(1) Distribution');
%set x and y axis limits
ylim([0, 1.5]);
xlim([0, max(x)]);
legend('Location', 'Best');
grid on;

hold off;
```

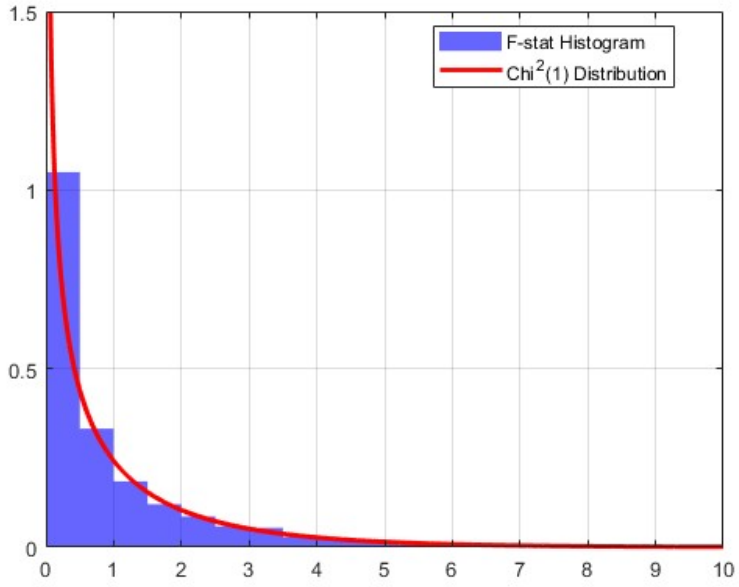


Figure 13: Histogram of the F-statistics, plot of $\chi^2(1)$

Deterministic time trend

In this last section we perform a DF test on a deterministic time trend process. First, we generate the deterministic time model and the test statistics via the command `adftest`.

```
simu = 1000;
T = 250;  % Number of data points in each simulation
```

```
3
4 for j = 1:simu
5     beta = 0.5;  % Set the time trend coefficient
6     epsilon = randn(T, 1);  % Generate white noise
7
8     % Create a time series with a deterministic time trend
9     t = (1:T)';
10    y = beta * t + cumsum(epsilon);
11
12    % Perform the Dickey-Fuller test with the correct distribution
13    [h, pValue,stat] = adftest(y, 'lags',1 , 'model', 'TS');
14    emp_DF(j) = stat;
15 end
```

Having obtained the test statistics, we can see that their empirical distribution closely resembles a Dickey and Fuller distribution in the case of inclusion of both a constant term and a deterministic time component. This distribution is different from the one we found in excercise 8.

```
1 histogram(emp_DF, 50, 'Normalization', 'probability')
2 xlabel('DF-stat')
3 ylabel('Probability')
4 title('Empirical␣Distribution␣DF-Stat')
```

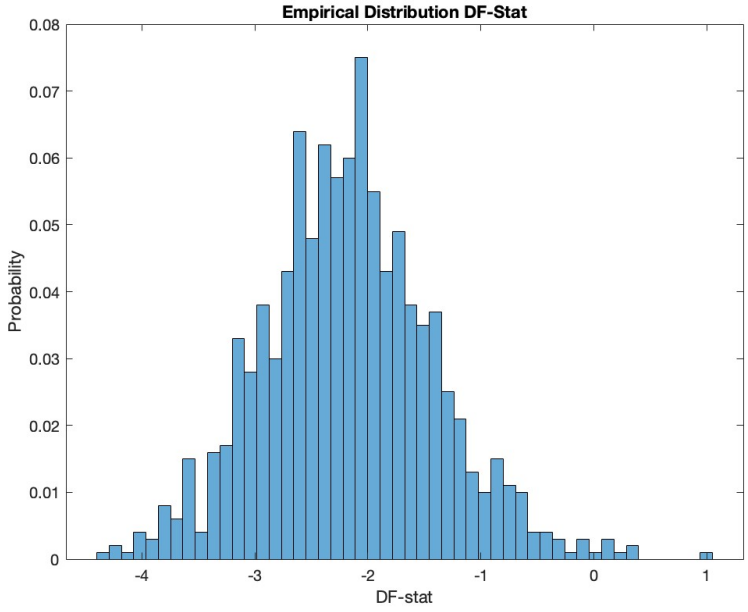


Figure 14: empirical distribution of the test statistic

We can more formally assess this correspondence by looking at few percentiles of our empirical distribution

```
percentiles = prctile(emp_DF, [1,2.5,5,10])
```

which gives values almost identical to the ones tabulated by DF in the case of a differenced regression with both a constant and a time trend component. $(-3.99, -3.69, -3.43, -3.13)$.
We can further verify the correctness of implementing a DF-test by checking the rejection rate of a 0.05-level test using the above-mentiod critical threshold

```
sum(emp_DF < -3.43)/simu
```

which indeed tells us we have wrongly rejected the null hypothesis 5% of the times.