Empirical exercise – Violation of the spherical errors assumption with serial correlation

1. Aim of the exercise

To understand the implications of violating the spherical errors assumption with serial correlation for the sampling distribution of the OLS estimator using simulation.

2. Set a seed for reproducible results

Set a seed for reproducible results.

```
clear;
rng(1)
```

3. Set the number of simulations

Set the number of simulations to be carried out.

```
N_sim = 1000;
```

4. Set the sample size

Assume that we have a linear regression model that contains a constant term and an independent variable. In this simulation exercise we will consider a time-series specification for the errors of the model. Hence, we set the sample size to 50 in this exercise because this is more typical of time-series models in social science (e.g., 50 years of annual data).

```
N_obs = 50;
```

5. Set true values for the coefficients of the intercept and the independent variable

Assume that we know the true values of the coefficients of the variables of the linear regression model we consider, and that these values are as indicated at the end of the section.

```
B_true = [0.2 0.5]';
N_par = 3;
```

6. Generate data for the systematic component of the regression model

Generate data for the systematic component of the regression model.

```
x_0 = ones(N_obs,1);
x_1 = unifrnd(-1,1,N_obs,1);
```

7. Create an empty matrix for storing the simulated OLS coefficient estimates

The code presented at the end of the section creates an empty matrix. The empty matrix is to store in each of its columns the `N_sim` simulated coefficient estimates from three different regression models. Therefore, the matrix is `N_sim` $\times$ `N_par`, where `N_par` is the number of regression models.

```
B_hat_x_1_sim = NaN(N_sim,N_par);
```

8. Create the sampling distribution of the OLS estimator

A problem that can arise with an OLS model is serial correlation where one or more of a model's errors influence one or more other errors. This is most often considered in the context of time-series analysis where past values of the error term influence current values.

The presence of serial correlation by itself is an efficiency problem, similar to that with heteroskedasticity. MATLAB has the `arima` function for time-series analysis which we can use to simulate a time-series process. In particular, we will use the function to create an AR(1) process, which is serial correlation in which the value of the error term in the last period influences the current value, but values beyond one period back do not have any additional unique effect on the current value.

In the for loop presented at the end of the section, we conduct a simulation to create sampling distributions for the OLS estimator under three different model specifications. In the first model we assume that the errors of the model are normal.

In the second model we assume that the errors are AR(1). Here we assume an auto-correlation parameter value of 0.75 (it can vary between 0 and 1, with larger values indicating stronger serial correlation).

There are many possible solutions and strategies for dealing with serial correlation, the choice of which should depend on what you understand the underlying DGP to be. In the third model, we focus on the popular modelling strategy of including a lagged value of the dependent variable in the model as an independent variable.

```
for i = 1:N_sim
    model = arima('Constant',0,'AR',0.00,'MA',0,'Distribution','Gaussian','Variance',1);
    u = simulate(model,N_obs);
    X = [x_0 x_1];
    y = X*B_true+u;
    LSS = exercisefunctionlss(y,X);
    B_hat_x_1_sim(i,1) = LSS.B_hat(2,1);
    model = arima('Constant',0,'AR',0.85,'MA',0,'Distribution','Gaussian','Variance',1);
    u_ar = simulate(model,N_obs);
    X = [x_0 x_1];
    y_ar = X*B_true+u_ar;
```

```
    LSS = exercisefunctionlss(y_ar,X);
    B_hat_x_1_sim(i,2) = LSS.B_hat(2,1);
    y_ar_lag = lagmatrix(y_ar,1);
    X = [x_0 x_1 y_ar_lag];
    LSS = exercisefunctionlss(y_ar(2:N_obs,:),X(2:N_obs,:));
    B_hat_x_1_sim(i,3) = LSS.B_hat(2,1);
end
```

9. Plot serially correlated and normal errors

Plot serially correlated and normal errors.

```
plot(u)
hold on
plot(u_ar)
title('Fig 1.  Errors that are serially correlated and normal')
legend('Errors are normal','Errors are serially correlated')
hold off
```

10. Plot the sampling distribution of the OLS estimator when errors are serially correlated and when they are not

Plot the sampling distributions of the OLS estimator when errors are serially correlated and when they are not. What do you conclude?

```
ksdensity(B_hat_x_1_sim(:,1))
hold on
ksdensity(B_hat_x_1_sim(:,2))
title('Fig 2.  Sampling distribution of the OLS estimator')
legend('Errors are normal','Errors are serially correlated')
ylabel('Kernel smoothed density')
xlabel('B_hat_x_1_sim')
hold off
```

11. Plot the sampling distribution of the OLS estimator when lagged dependent is included in the model and when it is not included

The plot gives the distribution of estimates when the lagged dependent is included in the model and when it is not. The result is a trade-off between bias and efficiency. The estimator of the model that includes the lagged dependent variable is more efficient: the peak of the distribution of the estimated values is higher, and the distribution is less spread compared with the distribution of the estimates for the model that does not include the lagged dependent variable. However, also notice that its peak is slightly off from the true value. In contrast, the distribution of the estimates from the OLS regressions that did not include the lagged depen-

dent variable as an independent variable is centered right on 0.50. However, the spread of the estimates for this model is larger, indicating less efficiency. This shows that including a lagged value of the dependent variable as an independent variable in the model improves efficiency because it captures serial correlation in the residual, but at the cost of some bias. This bias emerges because the lagged value of the dependent has a random component to it. That is, by definition, the dependent variable consists of a systematic and a stochastic component that leads to some level of bias.

```
ksdensity(B_hat_x_1_sim(:,2))
hold on
ksdensity(B_hat_x_1_sim(:,3))
line([mean(B_hat_x_1_sim(:,2)) mean(B_hat_x_1_sim(:,2))],ylim,'Color','blue')
line([mean(B_hat_x_1_sim(:,3)) mean(B_hat_x_1_sim(:,3))],ylim,'Color','red')
title('Fig 3.  Sampling distribution of the OLS estimator')
legend('Lagged dependent not included','Lagged dependent included','Mean of B_hat_x_1_si
when lagged dependent not included','Mean of B_hat_x_1_sim when lagged dependent
included')
ylabel('Kernel smoothed density')
xlabel('B_hat_x_1_sim')
```