Exercise – Understanding indirect inference

1. Aim of the exercise

The aim of this exercise is to apply indirect inference as a simulation-based estimation technique for recovering parameters in models with analytically intractable likelihood functions. Specifically, we focus on estimating the parameter of a moving average process using an auto regressive auxiliary model. Through repeated simulations, auxiliary model fitting, and optimization, we demonstrate how indirect inference can effectively approximate true model parameters and assess their statistical significance.

2. Theory

Indirect inference is a method for estimating the parameters of a complex model when its likelihood function is difficult or impossible to work with analytically. This typically occurs in models with nonlinear relationships, unobserved variables, or missing data, where the likelihood function is either intractable or computationally prohibitive. Rather than estimating the parameters directly, the method uses simulation and comparison to approximate them.

Consider a data-generating process referred to as a complex model, also called a structural or economic model, whether derived from formal theory or specified empirically. This model is defined as

$$y_t = G(y_{t-1}, x_t, u_t; \theta),$$

where $\{y_t\}_{t=1}^T$ are observed endogenous variables, $\{x_t\}_{t=1}^T$ are observed exogenous variables, and $\{u_t\}_{t=1}^T$ are unobserved random errors. The initial condition y_0 is known, and the errors u_t are assumed to be i.i.d. according to a known distribution F. The goal is to estimate the parameter vector $\theta \in \mathbb{R}^k$ that governs this model.

To facilitate estimation, indirect inference introduces an auxiliary model, a simplified statistical representation that is easier to estimate and designed to capture salient features of the data. This model is specified as

$$y_t = h(y_{t-1}, x_t; \beta) + \varepsilon_t,$$

where the error term ε_t follows a known distribution: $\varepsilon_t \sim F_\varepsilon$. The corresponding conditional density function is given by

$$f_\varepsilon(y_t - h(y_{t-1}, x_t; \beta)) = f(y_t \mid y_{t-1}, x_t; \beta),$$

with $\beta \in \mathbb{R}^p$ denoting the auxiliary parameter vector. It is required that $p \geq k$, ensuring that the auxiliary model possesses sufficient flexibility to reflect the behavior of the complex model. Although the auxiliary model may be misspecified, its purpose is not to replicate the structural process exactly, but rather to provide a tractable summary of the key statistical features present in the data.

The auxiliary parameters $\hat{\beta}$ are estimated from the observed data by maximizing the log-likelihood function

$$\hat{\beta} = \arg\max_\beta \sum_{t=1}^T \log f(y_t \mid y_{t-1}, x_t; \beta),$$

yielding a statistical summary of the observed data. Simulated data are then generated from the complex model using trial values of θ. The auxiliary model is re-estimated on each simulated

dataset to obtain

$$\tilde{\beta}(\theta) = \arg\max_{\beta} \sum_{t=1}^{T} \log f(\tilde{y}_t(\theta) \mid \tilde{y}_{t-1}(\theta), x_t; \beta),$$

where $\tilde{y}_t(\theta)$ denotes the simulated data generated under parameter θ. The core idea is to adjust θ so that the auxiliary estimates from the simulated data, $\tilde{\beta}(\theta)$, closely match those from the observed data, $\hat{\beta}$.

If the model is exactly identified ($p = k$), it may be possible to find a θ that reproduces $\hat{\beta}$ exactly. In most cases, however, the model is overidentified ($p > k$), and a distance metric is used to measure the distance between $\tilde{\beta}(\theta)$ and $\hat{\beta}$. Likelihood-based estimation of the auxiliary parameters ensures consistency and efficiency, making the comparison between observed and simulated data statistically meaningful.

In summary, indirect inference estimates the parameters of a complex model by leveraging a simpler auxiliary model. The auxiliary model serves as a lens through which the statistical behavior of the observed data is summarized and compared to simulated data. By iteratively adjusting θ to minimize the distance between auxiliary estimates from observed and simulated data, the method yields an indirect estimate of the structural parameters governing the complex model.

Indirect inference possesses well-defined asymptotic properties that support its use in econometric analysis. Under standard regularity conditions, the estimator of the parameters is consistent, meaning it converges in probability to the true parameter value as the sample size increases. Furthermore, the estimator is asymptotically normal, which allows researchers to construct confidence intervals and perform hypothesis tests using standard statistical tools. These properties hold even though the auxiliary model may be misspecified, as long as it captures relevant features of the data and the mapping from structural to auxiliary parameters is sufficiently informative. The asymptotic behavior is driven by the sample size, not by the number of simulations, which primarily affects the precision of the estimator in finite samples.

3. True and auxiliary model setup for indirect inference

In this exercise, we assume that the true data-generating process is an MA(1) model, given by the form $y_t = \epsilon_t + \theta\epsilon_{t-1}$, where ϵ_t is white noise. Estimating the MA(1) parameter is often challenging due to the complexity of the likelihood function and the risk of non-invertibility (Osborn, 1976). To address these difficulties, we adopt an indirect inference approach, using an AR(1) model as an auxiliary representation. The AR(1) process is specified as $y_t = \beta y_{t-1} + \epsilon_t$, and the estimated coefficient β serves as a summary statistic to match simulated and observed data.

4. Set simulation parameters

Clear the workspace. Set the number of simulated datasets to 100. This means the model will generate 100 different datasets by repeatedly simulating data with different random errors. Averaging across these simulations reduces the impact of random variation in any single dataset, leading to more stable and reliable estimates. Specify the true value of the moving average parameter as 0.5. This is the target value we aim to recover through indirect inference. Specify the standard deviation of the error term as 1. These values represent the underlying structure of the model that generates the data. Set the length of each time series to 1000 observations, meaning each simulated dataset will contain 1000 data points. Finally, define a

weight matrix, which in this case is simply the number 1. This matrix will later be used to calculate how far the simulated results deviate from the observed data when estimating the model parameters using indirect inference.

```
%% 4. Set simulation parameters

% 4.1. Clear workspace
clear;

% 4.2. Number of simulated datasets
N_sim = 100;

% 4.3. True MA(1) parameter (structural)
theta_true = 0.5;

% 4.4. Standard deviation of error term
sigma_true = 1;

% 4.5. Length of time series
T = 1000;

% 4.6. Weight matrix for objective function (scalar)
W = 1;
```

5. Simulate MA(1) process

This section simulates the MA(1) process. Begin by generating a sequence of random error terms from a normal distribution with a specified mean and standard deviation. These error terms form the basis of the time series. Next, initialize an empty vector to store the simulated values and fill it using a loop that applies the MA(1) formula: each value in the time series is calculated as the sum of the current error term and a scaled version of the previous error term, where the scaling factor is the true ma parameter. The result is a synthetic time series that reflects short-term dependencies, which will later be used to estimate parameters through indirect inference.

```
%% 5. Simulate MA(1) process

% 5.1. Generate error terms
epsilon = random("Normal",0,sigma_true,[T+1,1]);

% 5.2. Initialize time series
y = NaN(T,1);

% 5.3. Generate MA(1) data using theta_true
for t = 2:T+1
    y(t-1) = epsilon(t)+theta_true*epsilon(t-1);
end
```

6. Plot simulated data

Create a visual representation of the simulated MA(1) time series.

```
%% 6. Plot simulated data

% Plot
figure
plot(y)
title('Fig. 1. Simulated MA(1) process');
xlabel('Time');
ylabel('y_t');
```

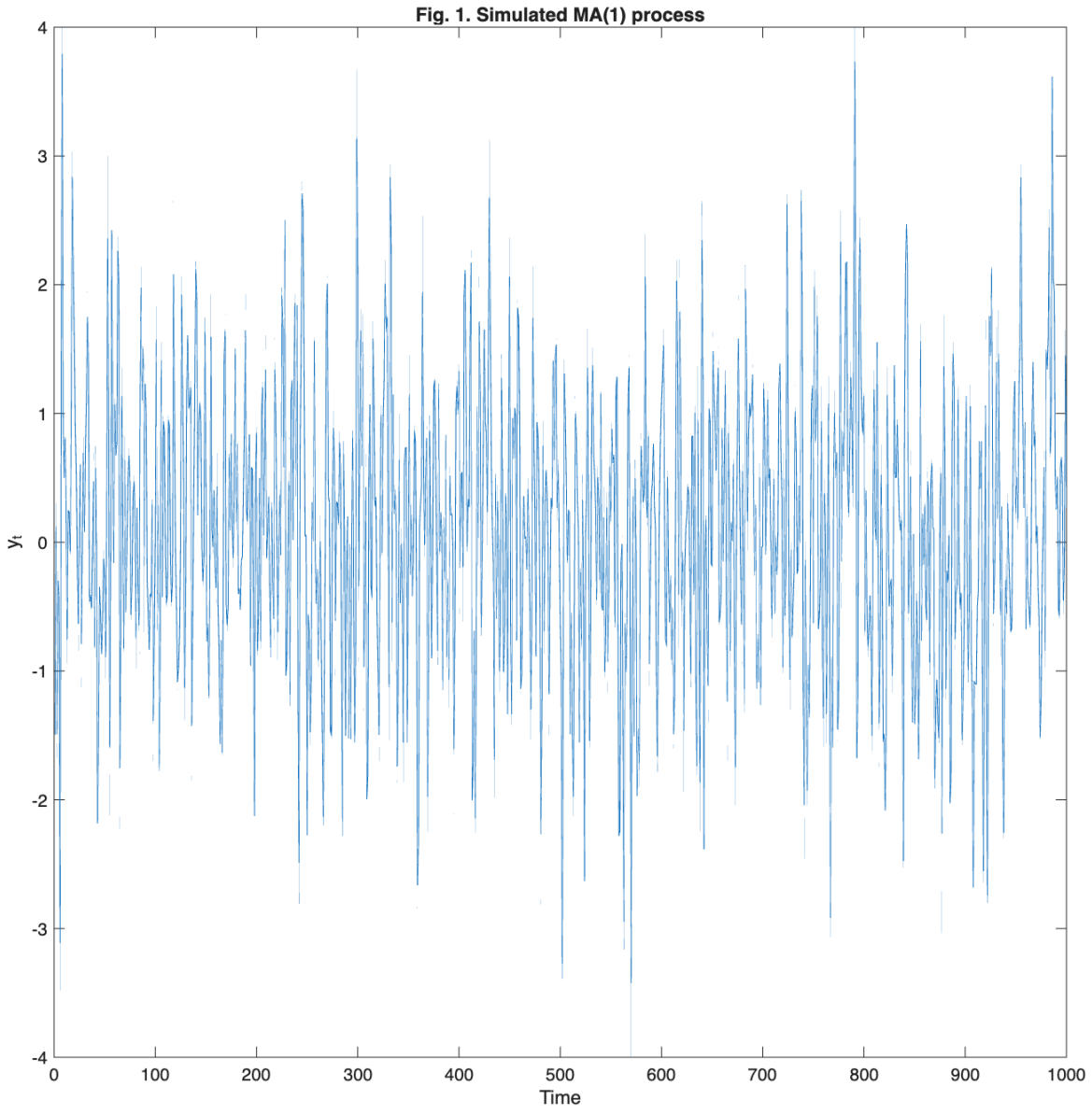
Fig. 1. Simulated MA(1) process
4
3
2
1
0
-1
-2
-3
-4
y_t
0
100
200
300
400
500
600
700
800
900
1000
Time

7. Define auxiliary model estimation function

Refer to the PDF file on the auxiliary model.

```
%% 7. Define auxiliary model estimation function
% See the function file auxiliary.m.
```

8. Estimate auxiliary model on simulated data

Estimate the parameter of the auxiliary model using observed data.

```
%% 8. Estimate auxiliary model on observed data

% AR(1) estimate from MA(1) data
beta_hat = auxiliary(y);
```

9. Define indirect inference function

We define an indirect inference function to minimize a distance measure between auxiliary estimates. The function has five input arguments: a candidate value for the structural parameter θ used to simulate data from the MA(1) model, the number of simulations to perform, the length of the time series, the diagonal weight matrix W used to compute the distance measure Q, and the auxiliary parameter $\hat{\beta}$ estimated from the observed data. For each simulation iteration, we generate data from the true data-generating process, in this case, the MA(1) model, using the candidate value of θ. We then estimate the auxiliary parameter β from each simulated dataset using the AR(1) model. The function computes the average of these simulated estimates, denoted $\tilde{\beta}(\theta)$, and calculates the distance measure Q. Since we aim to find a value of θ that makes the auxiliary estimates from simulated data as close as possible to those from the observed data, we define the objective function as: $Q(\theta) = (\hat{\beta} - \tilde{\beta}(\theta))'W(\hat{\beta} - \tilde{\beta}(\theta))$.

```
%% 9. Define indirect inference objective function
% See the function file objective.m.
```

10. Estimate the parameter of the complex model using indirect inference

This section performs the core estimation step in the indirect inference procedure, targeting the MA(1) parameter. It begins by specifying an initial guess for the parameter value, which serves as the starting point for the optimization algorithm. The optimization settings are configured using optimoptions, selecting the Sequential Quadratic Programming method and setting tolerances and iteration limits to control the precision and efficiency of the search. The `fmincon` function is then used to minimize the objective function, which measures the distance between the AR(1) estimate derived from the observed data and the average AR(1) estimate obtained from simulated datasets generated using candidate values of the MA(1) parameter. The result is an estimated value of θ that best aligns the simulated model with the empirical data, providing an indirect estimate of the underlying MA(1) process.

```
%% 10. Estimate the parameter of the complex model using ind. inf.

% 10.1. Initial guess for optimal theta
```

```
theta_initial = 0.5;

% 10.2. Optimization output
options = optimoptions('fmincon', ...
    'Algorithm','sqp', ...
    'Diagnostics','off', ...
    'Display','iter-detailed', ...
    'MaxIterations',100, ...
    'MaxFunctionEvaluations',100, ...
    'OptimalityTolerance',1e-4, ...
    'StepTolerance',1e-4, ...
    'SpecifyObjectiveGradient',false);

% 10.3. Optimize to find theta that minimizes the objective function Q
theta_estimated = fmincon(@(theta) ...
    objective(theta,N_sim,T,W,beta_hat),theta_initial, ...
    [],[],[],[],[],[],[],options);
```

11. Simulate auxiliary estimates for Wald test preparation

This section performs a Monte Carlo simulation to generate repeated estimates of an auxiliary model parameter, likely an AR(1) coefficient, using data simulated from an MA(1) process. Each iteration creates a new sequence of random error terms, constructs a time series based on the estimated MA(1) parameter, and then re-estimates the auxiliary parameter from the simulated data. The resulting collection of estimates is stored for later analysis.

```
%% 11. Simulated estimation of AR(1) coefficients via MA(1) model

% 11.1. Preallocate vector to store AR(1) estimates from simulations
beta_hat_sim = NaN(N_sim,1);

% 11.2. Simulate data using estimated theta and re-estimate AR(1)
for i = 1:N_sim
    epsilon_sim = random("Normal",0,1,[T+1,1]);
    y_sim = zeros(T,1);
    for t = 2:T+1
        y_sim(t-1) = epsilon_sim(t)+theta_estimated*epsilon_sim(t-1);
    end
    beta_hat_sim(i) = auxiliary(y_sim);
end
```

12. Compute Wald statistic and critical Value

We calculate and evaluate the Wald statistic to test whether the estimated MA(1) parameter is statistically different from the true value used to generate the data. To do this, we first compute the variance of the auxiliary AR(1) estimates obtained from the simulated datasets. Although these estimates come from an auxiliary model, their variability reflects the uncertainty in the estimation of the MA(1) parameter. This variance is used as the denominator in the Wald formula. The Wald statistic is then calculated by taking the squared difference between

the estimated and true MA(1) values, divided by the variance of the simulated estimates. The result is displayed for reference. Next, we define the degrees of freedom for the test, which is set to one since we are testing a single parameter. We then calculate the critical value from the chi-squared distribution at the 95 percent confidence level. This value serves as the threshold for determining statistical significance.

To interpret the result, we compare the calculated Wald statistic to the critical value. If the Wald statistic exceeds the critical value, we conclude that the estimated parameter is significantly different from the true value, and the code displays a message indicating that the null hypothesis is rejected. If the Wald statistic is less than or equal to the critical value, the difference is not statistically significant, and the code displays a message stating that the null hypothesis cannot be rejected. This outcome helps assess whether the estimation method has successfully recovered the true parameter.

```
%% 12. Compute Wald statistic and critical value

% 12.1. Variance of simulated AR(1) estimates
var_beta_hat = var(beta_hat_sim);

% 12.2. Wald statistic
wald_statistic = (theta_estimated-theta_true)^2/var_beta_hat;

% 12.3. Display Wald statistic
disp(['Wald statistic: ',num2str(wald_statistic)]);

% 12.4. Degrees of freedom
df = 1;

% 12.5. Critical value from chi-squared distribution
chisq_critical_value = chi2inv(0.95,df);

% 12.6. Hypothesis test
if wald_statistic > chisq_critical_value
    disp(['Reject the null hypothesis: ' ...
        'The parameters are significantly different.']);
else
    disp(['Fail to reject the null hypothesis: ' ...
        'The parameters are not significantly different.']);
end
```

13. Simulation stability and the shape of the objective function

Following the diagnostic plot code, we evaluate how the objective function behaves across a range of candidate values for the parameter of the complex model. For each value in the theta grid, the code simulates multiple datasets using that candidate parameter, estimates the auxiliary AR(1) coefficient from each simulation, averages those estimates, and then computes the distance between this average and the auxiliary estimate obtained from the original data. The resulting values form the curve shown in the plot.

The clarity and smoothness of this curve depend heavily on the number of simulations used. If the number is too low, say, ten, the average auxiliary estimate becomes noisy due to random

variation in the simulated datasets. This noise carries over into the objective function, making the curve jagged and potentially misleading. By increasing the number of simulations to 100, 500, or more, the averaging process becomes more stable, reducing the influence of randomness and producing a cleaner, more reliable curve. This smoother shape helps the optimizer identify the best-fitting parameter more accurately and ensures that the indirect inference procedure yields meaningful results.

```
%% 13. Simulation stability and the shape of the objective function

% 13.1. Range of candidate theta values
theta_grid = linspace(0,1,100);

% 13.2. Preallocate for objective values
Q_values = zeros(size(theta_grid));

% 13.3. Compute the AR(1) estimate from original MA(1) data
beta_hat = auxiliary(y);

% 13.4. Loop over theta candidates
for i = 1:length(theta_grid)
    Q_values(i) = objective(theta_grid(i),N_sim,T,W,beta_hat);
end
```

14. Plot

The plot illustrates the core mechanism of indirect inference. It shows how the objective function responds to different candidate values of the MA(1) parameter, with a clear minimum around 0.45. Although the true parameter used to generate the data is 0.5, the slight deviation in the estimated minimum may reflect simulation noise, sample variability, or the limitations of the auxiliary model. The smooth, convex shape of the curve suggests that a sufficiently large number of simulations was used, reducing randomness and enhancing the reliability of the estimation. Such diagnostic plots are essential in validating the stability and sensitivity of the indirect inference procedure, ensuring that the optimizer is guided by a well-behaved objective landscape.

```
%% 14. Plot

% Plot
figure
plot(theta_grid,Q_values);
title('Fig. 2. Objective function Q(\theta) vs. candidate \theta');
xlabel('\theta (candidate)');
ylabel('Q(\theta)');
```

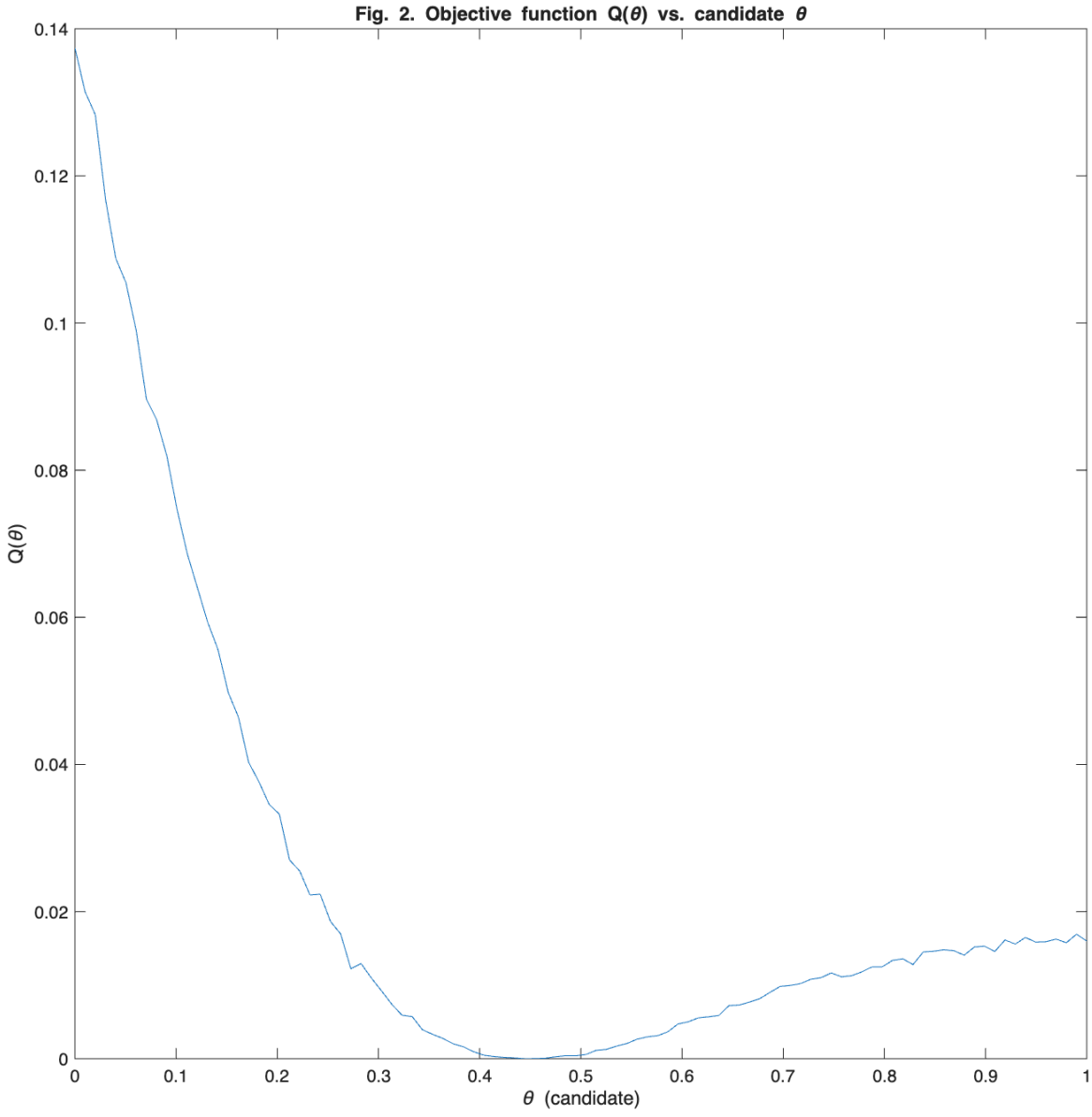
Fig. 2. Objective function Q(θ) vs. candidate θ
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
Q(θ)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
θ (candidate)

15. Final notes

This file is prepared and copyrighted by Simonas Stravinskas and Tunga Kantarcı. This file and the accompanying MATLAB file are available on GitHub and can be accessed using this link. This file has made use of Osborn, 1976. Maximum likelihood estimation of moving average processes. Annals of Economic and Social Measurement, 5 (1), 75-87.