14E026 Advanced Econometric Methods III

Empirical Problem 1

Suleman Dawood, Bjarni Einarsson, Adam Lee, Robertson Wang

1 Data exploration

1. Figure 1 plots the time series.

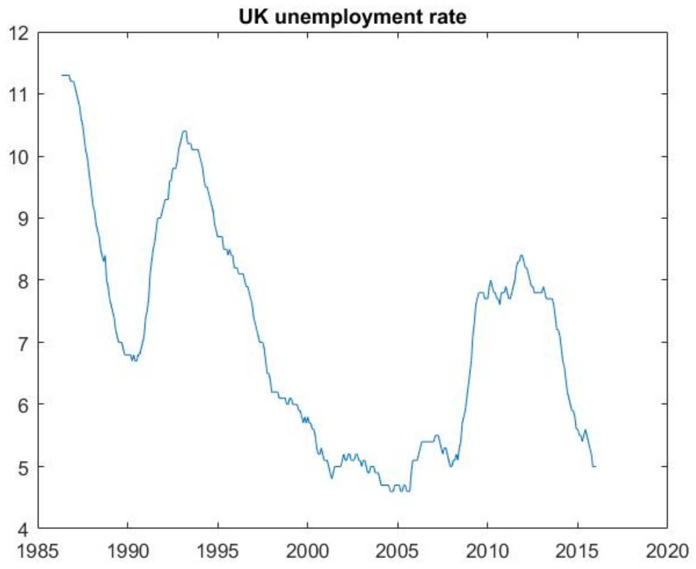

Figure 1: UK unemployment rate

2. Figure 2 plots the time series with gray bars indicating periods of recession.

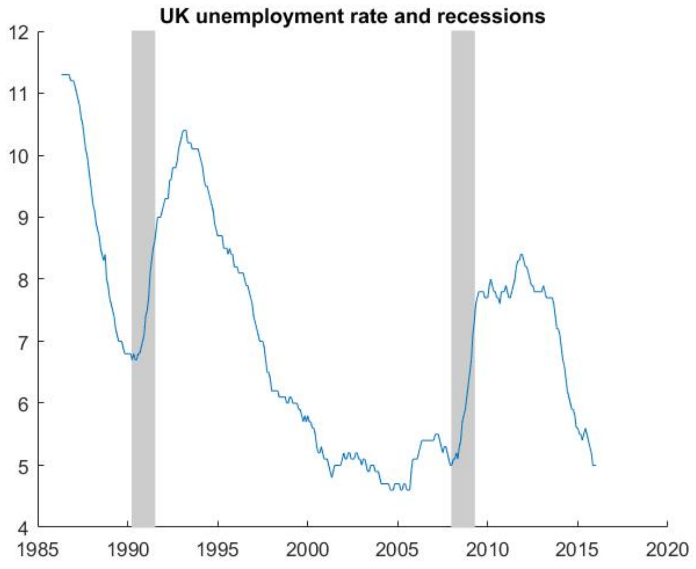

Figure 2: UK unemployment rate and recessions

3. The 1990s recession in the UK followed high inflation that was in part a result of a credit boom from continued stimulus (through lower taxes and interest rates) throughout the 1980s. One result of this stimulus policy was the UK's ill-fated entry into the European Exchange Rate Mechanism for a period

of 2 years. This policy culminated with “Black Wednesday”, when George Soros successfully forced the Bank of England to give up defending the value of Sterling, devalue, and leave the European Exchange Rate Mechanism. This was primarily a domestic event.

The late 2000s recession in the UK, was part of the “Great Recession” caused by the most recent financial crisis which emanated from the United States. However, the financial crisis ultimately impacted the global financial system making the recession a global event.

4. Figure 3 plots the autocorrelation function of our time series with 20 lags. This plot suggests that we do not have a pure MA process (unless $q > 20$), as the sample ACF does not drop off at any lag (before 20). We therefore expect $p > 0$, and cannot conclude anything about q.

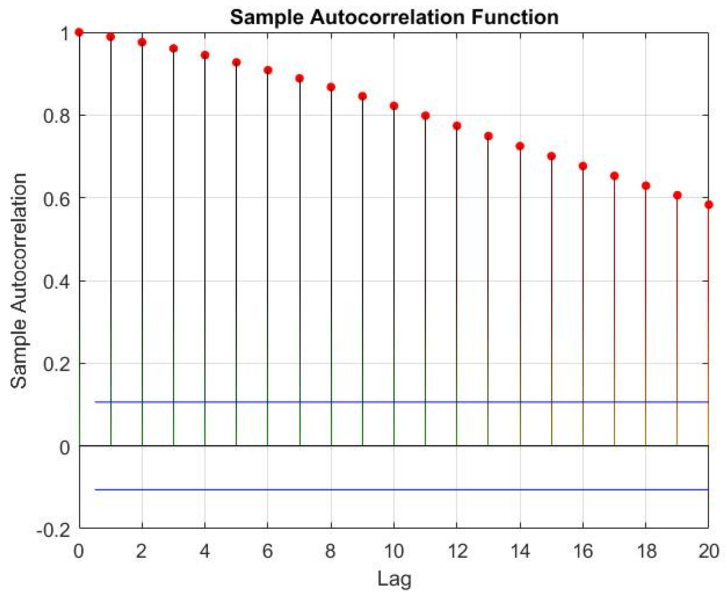

Figure 3: Sample ACF

5. Figure 4 plots the partial autocorrelation function of our time series with 20 lags. This figure suggests that we may expect an AR process of order 6, since most lags beyond this are not significant.

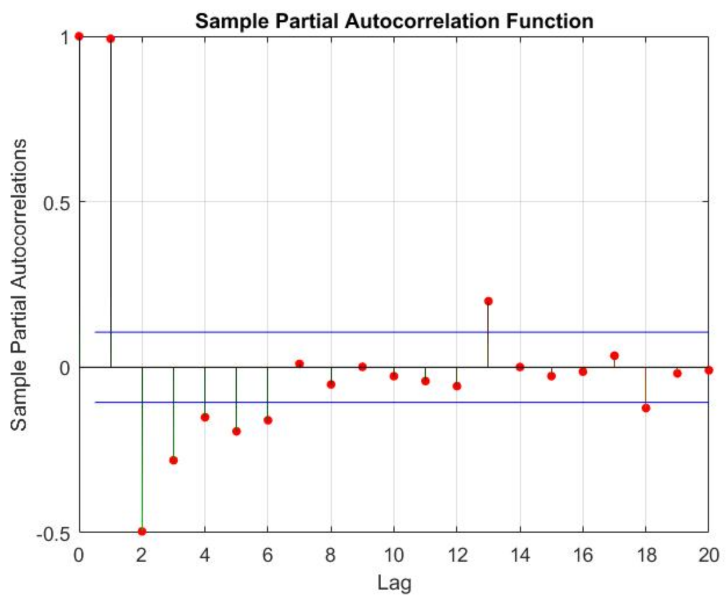

Figure 4: Sample PACF

6. Figure 5 plots the empirical density of our time series.

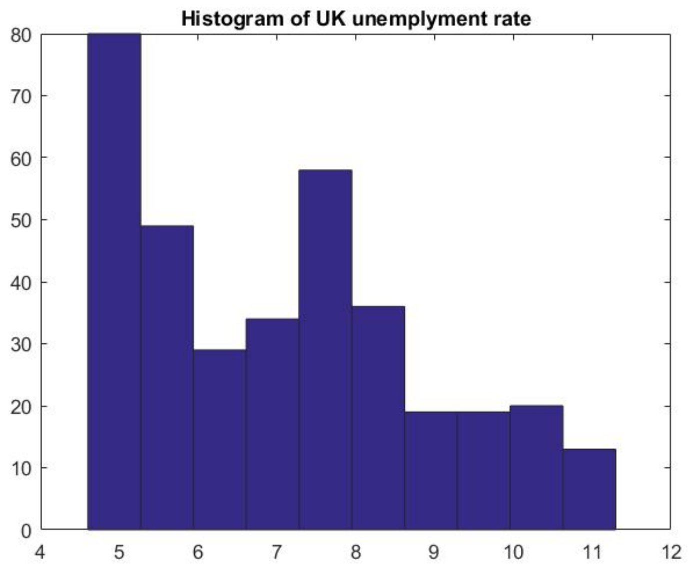

Figure 5: Empirical density

The plot suggests that the mean is somewhere around 7%, as we have a higher frequency in each bin within the shorter "tail" to the left of this demarcation, while a lower frequency in each bin but more bins overall to the right of it.

The variance of the plot is around 3. The spread is slightly larger than this to the right of the mean while it is slightly smaller to the left, weighted by the higher number of observations to the left.

The distribution has a positive skew (~ 0.47), describing the relatively higher concentration of observations to the left of the mean, and the long tail to the right.

The kurtosis of this data is around 2.2; this suggests it should come from a distribution that produces fewer and less extreme outliers than a normal distribution.

2 Models and Estimation

Note that an ADF test cannot reject the null hypothesis of a unit root in the data. Therefore, estimating an ARMA(p,q) model for the data in levels (as done below) is not theoretically justified. We believe this contributes to some of the more outlandish "results" we have obtained. One should begin by integrating the data until the null hypothesis of a unit root can be rejected. As we have not yet covered these techniques, we chose to follow the instructions in the pdf to the letter. Whilst our estimation routines seem to work well in Monte Carlo simulations when ϕ_1 is not close to 1, they start to perform much more poorly as ϕ_1 gets closer to 1. This is presented below. As a result, and since we cannot reject that $\phi_1 = 1$ we present results using both our estimation routine and the more robust routines built in to MATLAB. The results using the built-in routines are presented in sub-sections clearly marked as such.

2.1 Monte Carlo Simulations

We performed several Monte Carlo simulations to ensure that our estimation routines were working correctly. In the simulations below we used MATLAB to simulate 200 data points of the time series described, and then estimated the coefficients with our routine. This was performed 100 times for each MC simulation. The mean, median and plots of the estimated parameters are given below.

2.1.1 AR2 Model 1: $Y_t = 0.6Y_{t-1} + 0.2Y_{t-2} + \epsilon_t, \quad \epsilon_t \sim N(0, 0.04)$.

Table 1: Parameter estimates AR(2) Model 1

	c	ϕ_1	ϕ_2	σ^2
True	0	0.6	0.2	0.04
Mean	0.0006	0.5495	0.2282	0.0379
Median	0.0016	0.5414	0.2312	0.0380

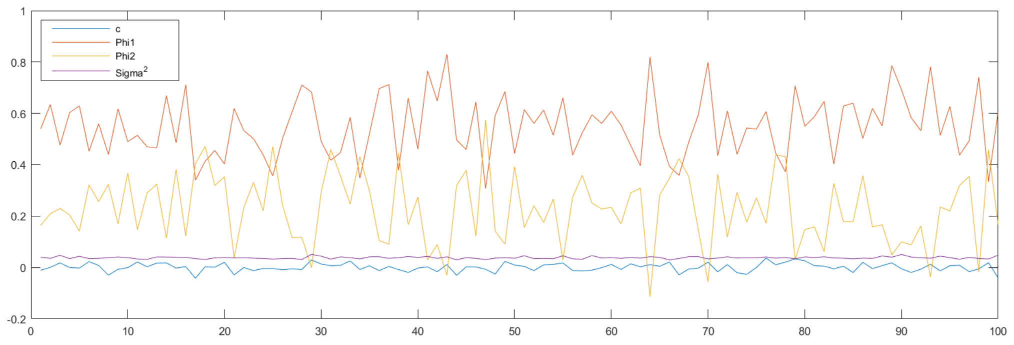

Figure 6: Parameter estimates for AR(2) Model 1

The estimation procedure is able to estimate this model without any issues.

2.1.2 AR2 Model 2: $Y_t = 0.8Y_{t-1} + 0.15Y_{t-2} + \epsilon_t$, $\epsilon_t \sim N(0, 0.04)$.

Table 2: Parameter estimates AR(2) Model 2

	c	ϕ_1	ϕ_2	σ^2
True	0	0.8	0.15	0.04
Mean	0.0054	0.7856	0.1400	2.3235
Median	0.0033	0.7913	0.1386	0.0383

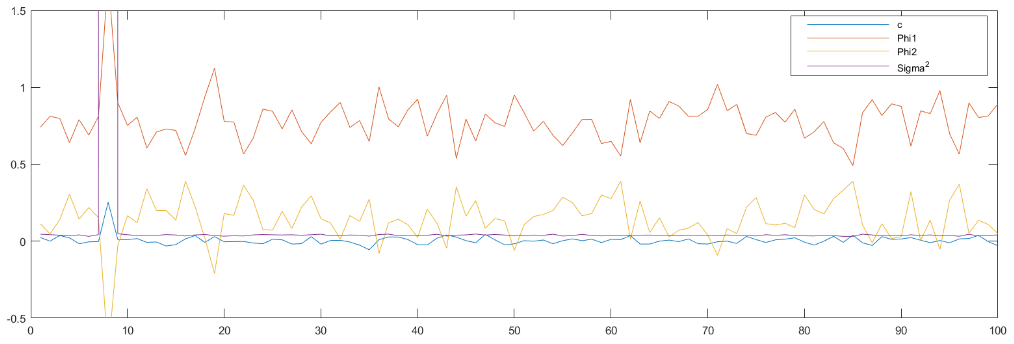

Figure 7: Parameter estimates for AR(2) Model 2

This is the first example we see of a recurring pattern with our estimation: as ϕ_1 gets closer to 1 and the data behaves more like it has a unit root, the parameter estimation experiences problems. In this simulation, as 0.8 is relatively far away from 1, there was only one such instance of this, with parameter estimates $c = 0.2543,\ \phi_1 = 1.7233,\ \phi_2 = -0.7289,\ \sigma^2 = 228.529$.

2.1.3 AR2 Model 3: $Y_t = 0.95Y_{t-1} + 0.03Y_{t-2} + \epsilon_t, \quad \epsilon_t \sim N(0, 0.04)$.

Table 3: Parameter estimates AR(2) Model 3

	c	ϕ_1	ϕ_2	σ^2
True	0	0.95	0.03	0.04
Mean	-0.1147	0.7680	-0.0321	19.9441
Median	-0.0051	0.9412	-0.0137	0.0393

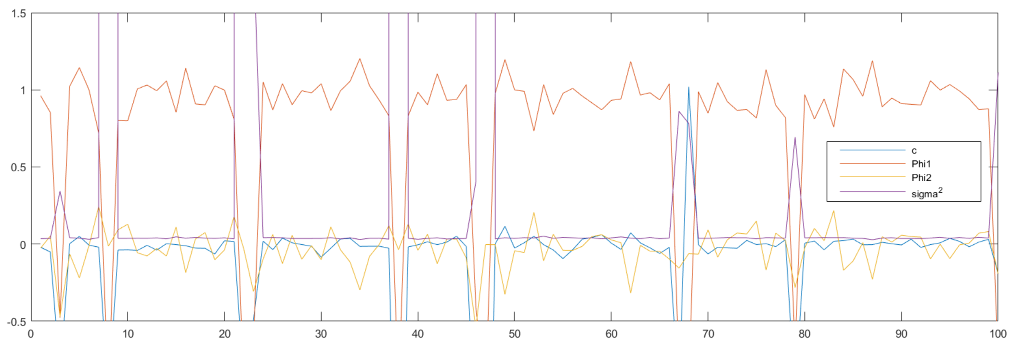

Figure 8: Parameter estimates for AR(2) Model 3

2.1.4 AR2 Model 4: $Y_t = 0.99Y_{t-1} + 0.005Y_{t-2} + \epsilon_t, \quad \epsilon_t \sim N(0, 0.04)$.

Table 4: Parameter estimates AR(2) Model 4

	c	ϕ_1	ϕ_2	σ^2
True	0	0.99	0.005	0.04
Mean	-0.0446	0.5930	-0.0569	64.5872
Median	0.0057	0.9430	-0.0348	0.0400

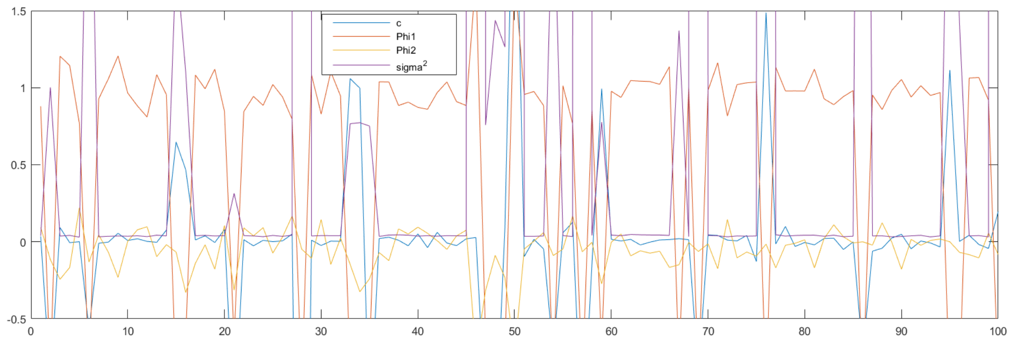

Figure 9: Parameter estimates for AR(2) Model 4

As can easily be seen from the last two plots, as ϕ_1 gets closer to 1, our estimation procedure experiences more incidences of poor performance.

2.1.5 ARMA(1,1) Model 1: $Y_t = 0.6Y_{t-1} + 0.2\epsilon_{t-1} + \epsilon_t, \quad \epsilon_t \sim N(0, 0.04)$.

Table 5: Parameter estimates ARMA(1,1) Model 1

	c	ϕ_1	θ_1	σ^2
True	0	0.6	0.2	0.04
Mean	0.0022	0.5749	0.2160	0.03942
Median	0.0040	0.5766	0.2330	0.0395

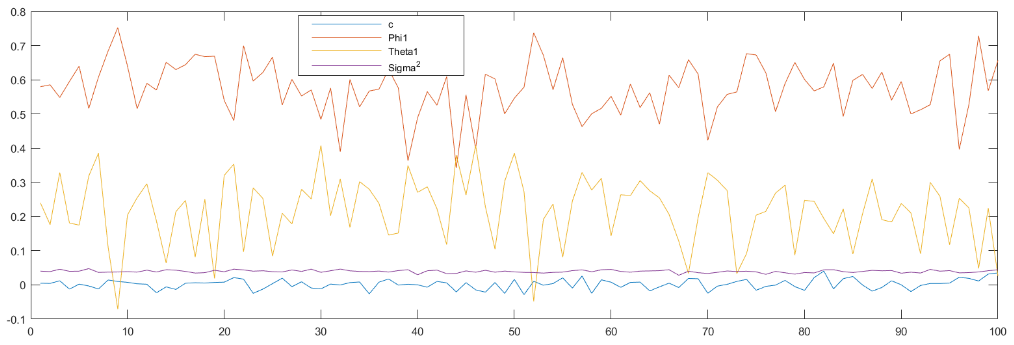

Figure 10: Parameter estimates for ARMA(1,1) Model 1

As with the AR(2) model, when ϕ_1 is far from 1, the routine does not experience significant problems when estimating the parameters.

2.1.6 ARMA(1,1) Model 2: $Y_t = 0.8Y_{t-1} + 0.15\epsilon_{t-1} + \epsilon_t, \quad \epsilon_t \sim N(0, 0.04)$.

Table 6: Parameter estimates ARMA(1,1) Model 2

	c	ϕ_1	θ_1	σ^2
True	0	0.8	0.15	0.04
Mean	0.0001	0.7824	0.1878	0.0389
Median	-0.0009	0.7901	0.1500	0.0393

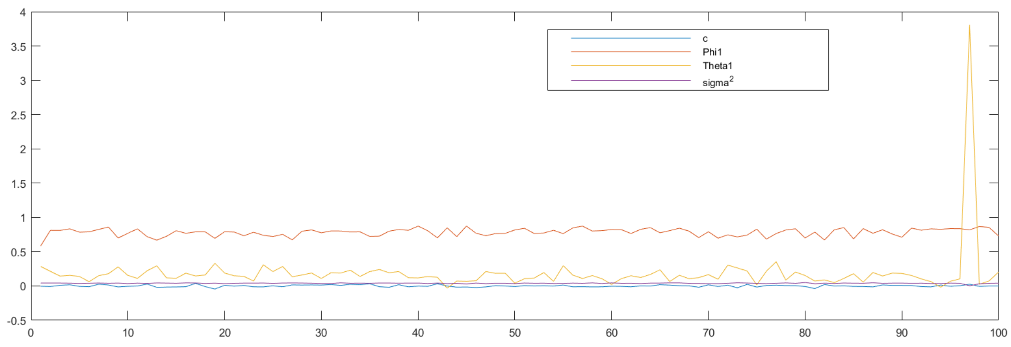

Figure 11: Parameter estimates for ARMA(1,1) Model 2

Here, again the estimation routine is generally fine, experiencing one problem during the simulation.

2.1.7 ARMA(1,1) Model 3: $Y_t = 0.95Y_{t-1} + 0.03\epsilon_{t-1} + \epsilon_t, \quad \epsilon_t \sim N(0, 0.04)$.

Table 7: Parameter estimates ARMA(1,1) Model 3

	c	ϕ_1	θ_1	σ^2
True	0	0.95	0.03	0.04
Mean	-0.0018	0.9217	5.9372	0.0308
Median	-0.0030	0.9301	0.0521	0.0377

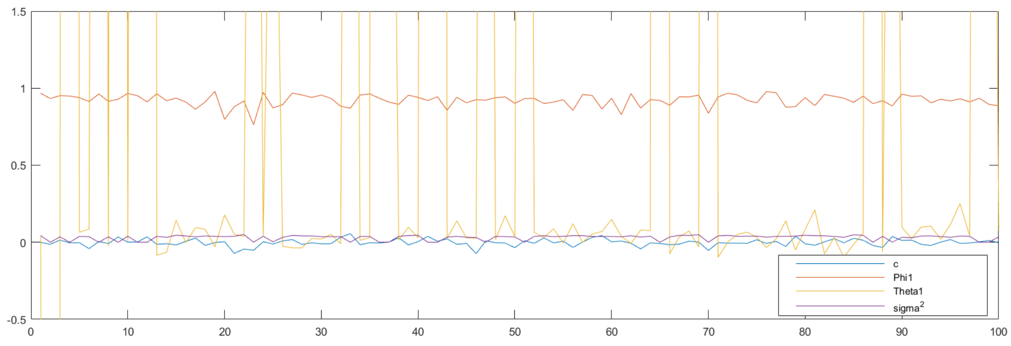

Figure 12: Parameter estimates for ARMA(1,1) Model 3

2.1.8 ARMA(1,1) Model 4: $Y_t = 0.99Y_{t-1} + 0.005\epsilon_{t-1} + \epsilon_t, \quad \epsilon_t \sim N(0, 0.04)$.

Table 8: Parameter estimates ARMA(1,1) Model 4

	c	ϕ_1	θ_1	σ^2
True	0	0.99	0.005	0.04
Mean	-0.0166	0.9404	8.5521	0.0449
Median	-0.0015	0.9636	0.0305	0.0387

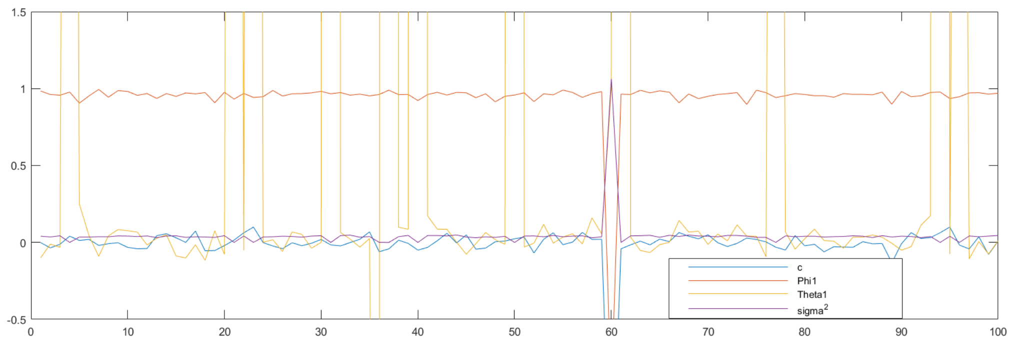

Figure 13: Parameter estimates for ARMA(1,1) Model 4

As in the AR(2) case, these plots reveal much more frequent estimation problems as ϕ_1 approaches 1.

2.2 ARMA models

Table 9: Parameter estimates AR(2) and ARMA(1,1)

	AR(2)		ARMA(1,1)	
	par.	se	par.	se
c	-0.0094	0.0079	10.1452	0.2423
ϕ_1	1.4587	0.0076	0.9890	0.0010
ϕ_2	-0.4588	0.0076	-	-
θ_1	-	-	-0.9937	0.0005
σ	0.0083	0.0038	2.0447	0.1374
l	-599.3647		363.0319	

1. Table 1 presents parameter estimates from the AR(1) and ARMA(1,1) models. The SEs were obtained as the square root of the diagonal elements of the inverse of the Hessian matrix computed by MATLAB during the optimisation routine.
2. We prefer the AR(2) model, due to the lower likelihood function (which we were minimising: it is the negative of the exact log-likelihood).
3. We observe that the residuals are not white noise, as would be expected for a stationary time series if the model were correctly specified.

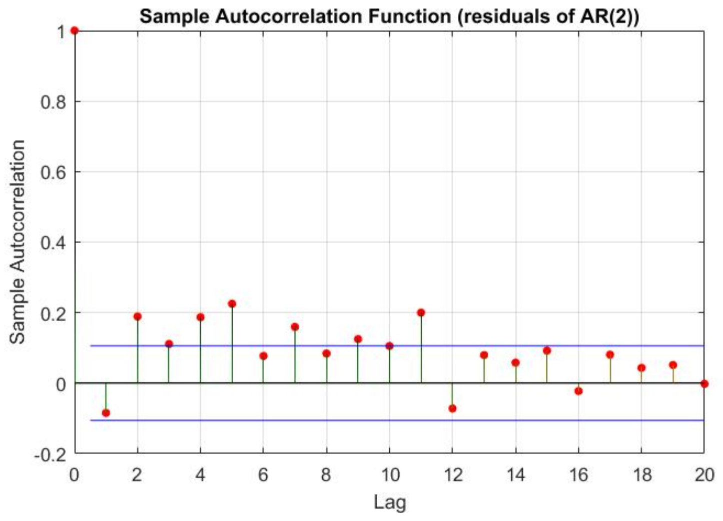

Figure 14: Sample ACF of AR(2) residuals

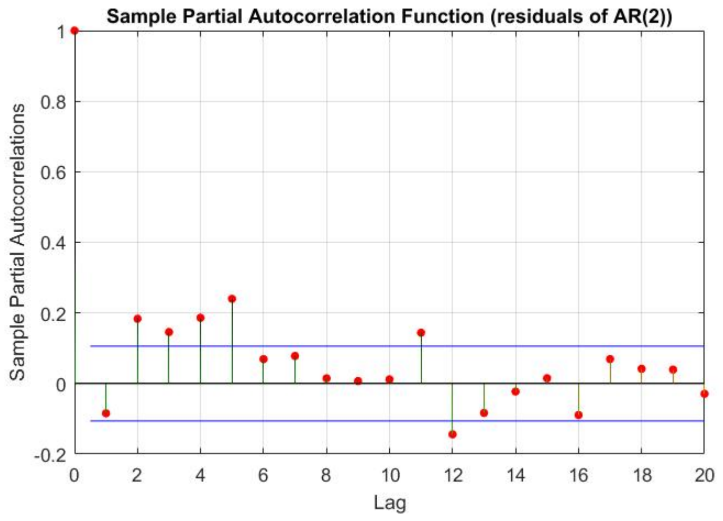

Figure 15: Sample PACF of AR(2) residuals

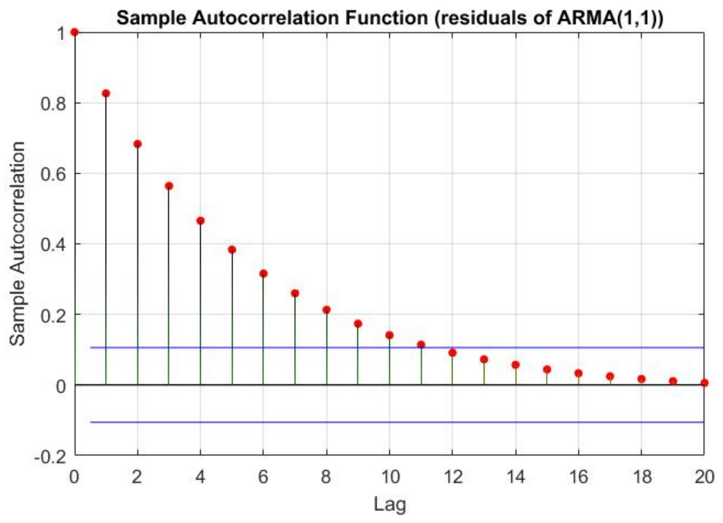

Figure 16: Sample ACF of ARMA(1,1) residuals

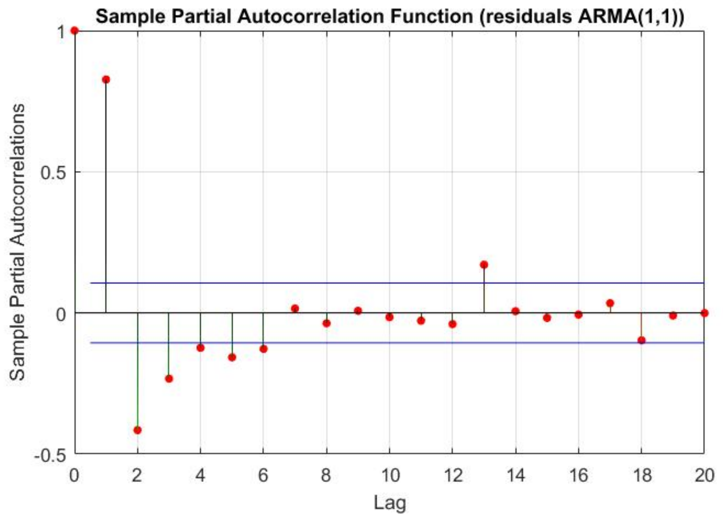

Figure 17: Sample PACF of ARMA(1,1) residuals

2.2.1 Using the built in function in Matlab

Table 10: Parameter estimates AR(2) and ARMA(1,1)

	AR(2)		ARMA(1,1)	
	par.	se	par.	se
c	0.03	0.0225	0.035	0.0319
ϕ_1	1.49161	0.0444	0.9927	0.0042
ϕ_2	0.49713	0.0444	-	-
θ_1	-	-	0.3364	0.0466
σ	0.0089	0.0005	0.0099	0.0006
l	337.0537		316.4257	

1. Table 10 presents parameter estimates from the AR(1) and ARMA(1,1) models using the inbuilt ARIMA function in Matlab. The standard errors are calculated using the same function.
2. We prefer the AR(2) model since it has a slightly higher likelihood and lower error variance. The models are very similar though and no clear way to choose between them.
3. We observe that the residuals are not white noise, as would be expected for a stationary time series if the model were correctly specified.

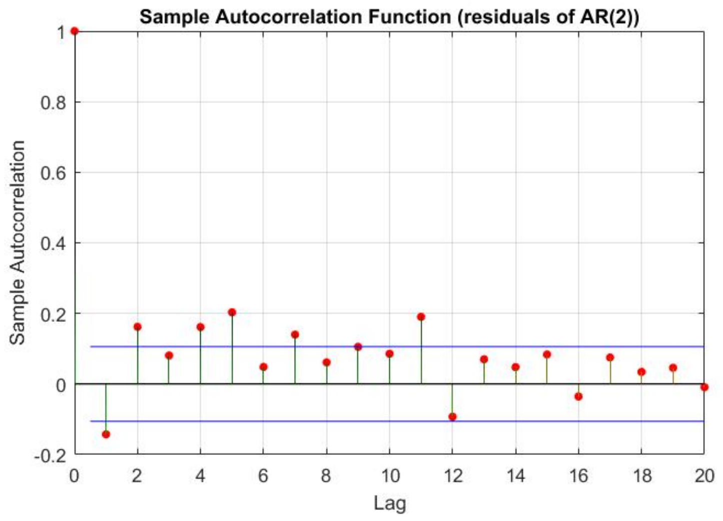

Figure 18: Sample ACF of AR(2) residuals

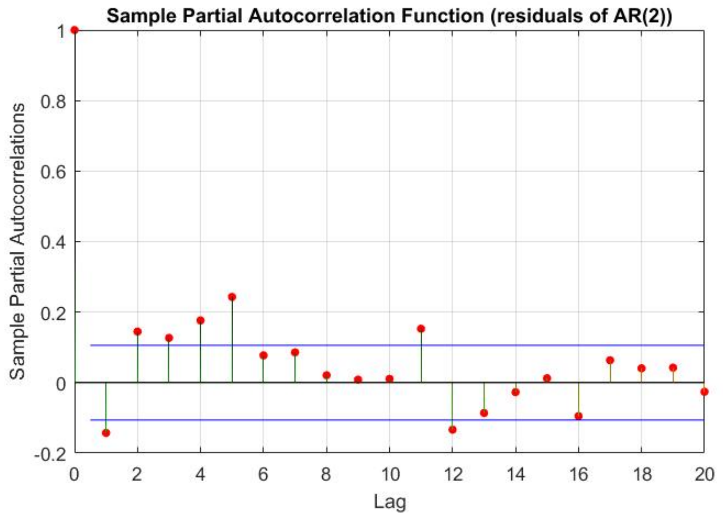

Figure 19: Sample PACF of AR(2) residuals

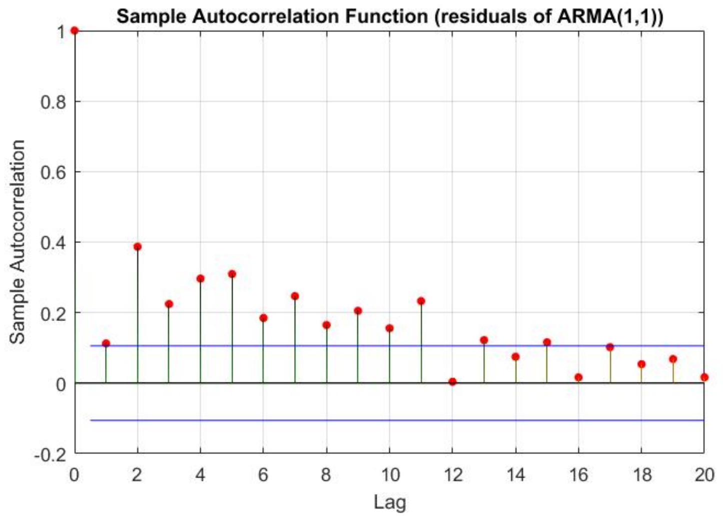

Figure 20: Sample ACF of ARMA(1,1) residuals

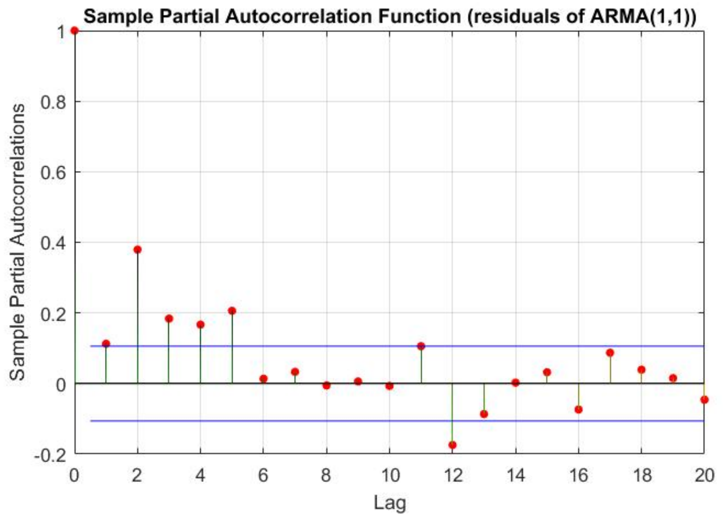

Figure 21: Sample PACF of ARMA(1,1) residuals

2.3 Model selection

We explored different lag structures using the AIC, AICC and BIC criteria. The AIC and AICC both select the AR(2) model while the BIC selects the ARMA(2,1). For reasons of parsimony we prefer the AR(2) model.

2.3.1 Using the built in function in Matlab

We explore combinations of lag structures for p and q from 0 to 5. We find that the AIC and AICC criteria prefer an ARMA(5,5) model with ARMA(4,3) having very close values. Short on their heels is the ARMA(2,1) model which we prefer for reasons of parsimony. The BIC criterion prefers very large lag structures but the ARMA(2,1) model is not too far off from the minimum value.

Table 11: AIC

	$q = 0$	$q = 1$	$q = 2$	$q = 3$	$q = 4$	$q = 5$
$p = 0$	-	1012	717.87	514.41	413.72	366.31
$p = 1$	-567.12	-626.85	-659.78	-664.11	-672.80	-695.61
$p = 2$	-668.11	-722.29	-720.34	-718.37	-720.12	-718.37
$p = 3$	-696.02	-720.33	-718.31	-719.25	-718.16	-722.18
$p = 4$	-702.37	-699.39	-681.14	-723.08	-721.42	-720.43
$p = 5$	-714.15	-702.81	-717.31	-715.75	-716.59	-723.47

Table 12: AICC

	$q = 0$	$q = 1$	$q = 2$	$q = 3$	$q = 4$	$q = 5$
$p = 0$	-	1012.47	717.93	514.52	413.89	366.55
$p = 1$	-567.09	-626.78	-659.66	-663.94	-672.56	-695.29
$p = 2$	-668.04	-722.18	-720.17	-718.13	-719.80	-717.96
$p = 3$	-695.91	-720.16	-718.07	-718.93	-717.75	-721.66
$p = 4$	-702.20	-699.15	-680.82	-722.67	-720.90	-719.79
$p = 5$	-713.91	-702.49	-716.90	-715.23	-715.96	-722.70

Table 13: BIC

	$q = 0$	$q = 1$	$q = 2$	$q = 3$	$q = 4$	$q = 5$
$p = 0$	-	1008.46	711.91	506.47	403.80	354.41
$p = 1$	-571.09	-632.80	-667.71	-674.03	-684.70	-709.49
$p = 2$	-674.06	-730.23	-730.26	-730.27	-734.00	-734.24
$p = 3$	-703.95	-730.25	-730.21	-733.14	-734.03	-740.03
$p = 4$	-712.29	-711.29	-695.03	-738.95	-739.27	-740.26
$p = 5$	-726.05	-716.70	-733.18	-733.60	-736.43	-745.29

Table 14: Parameter estimates of the preferred model

	ARMA(2,1)	
	par.	se
c	0.01541	0.00625
ϕ_1	1.93551	0.02313
ϕ_2	-0.93786	0.02301
θ_1	-0.69999	0.04642
σ	0.00757	0.00052
l	365.1466	

3 Forecasting

1. Figures 23 and 25 present the forecasts from both models (recall that we chose the AR(2) as our preferred model when using our estimation routine) together with the true outcome. As can be seen

below, our forecasts have issues which we believe are due to the problems our estimation routine has when ϕ_1 is close to 1, as demonstrated in the Monte Carlo section above.

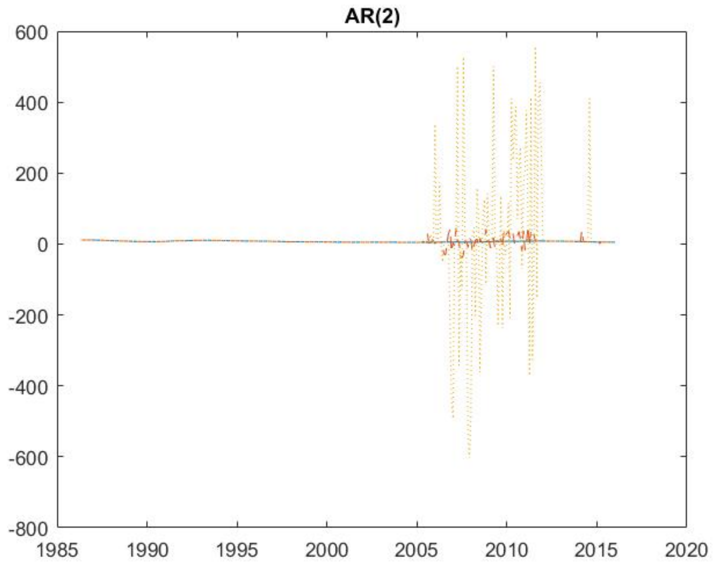

Figure 22: Forecast using AR(2)

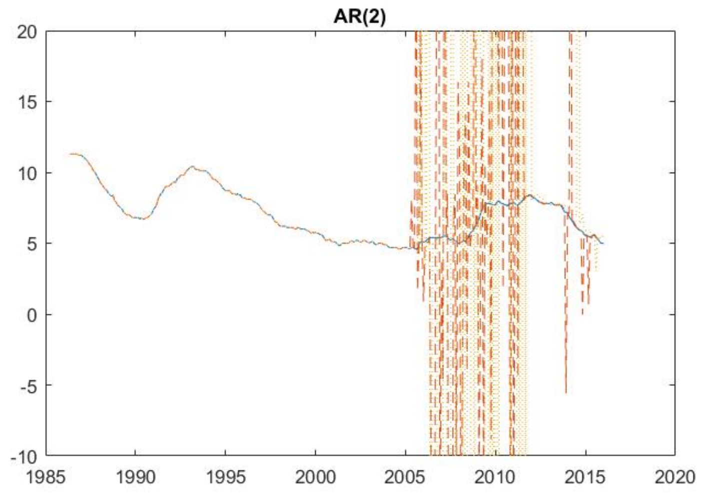

Figure 23: Forecast using AR(2) (trimmed)

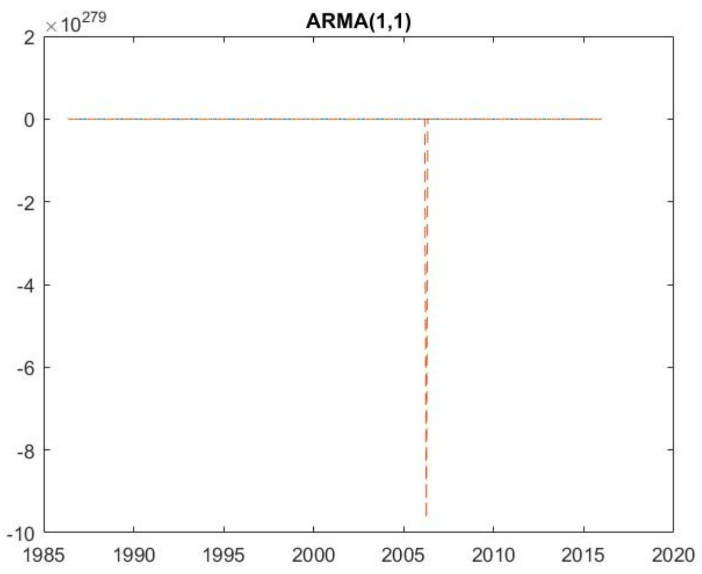

Figure 24: Forecast using ARMA(1,1)

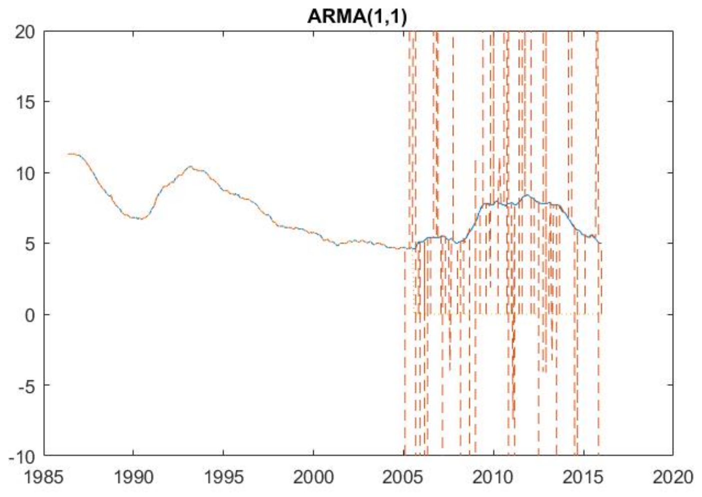

Figure 25: Forecast using ARMA(1,1) (trimmed)

2. Table 17 presents the mean-squared-forecast-error (MSFE) and mean-absolute-forecast-error (MAFE) for the 1 month ahead and the 6 months ahead forecast horizons.

Table 15: Forecast errors

	MSFE(1)	MAFE(1)	MSFE(6)	MAFE(6)
AR(2)	222.5140	8.5489	39858	111.3616
ARMA(1,1)	Inf	Inf	44.7673	6.5794

3. Based on the results in the table (and with all aforementioned caveats) we would prefer the AR(2)

model for a 1 month ahead forecast since it has lower MSFE and MAFE than the ARMA(1,1) model. For but a 6 months ahead forecast, on the other hand, the ARMA(1,1) model has lower MSFE and MAFE than the AR(2) model and would thus be preferred.

3.1 Using the built in function in Matlab

1. Figures 26, 27, and 28 present the forecasts from the models together with the true outcome.

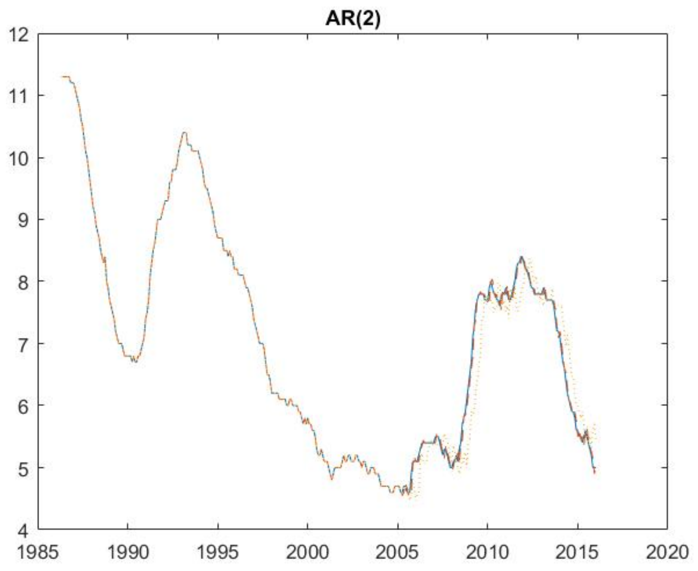

Figure 26: Forecast using AR(2)

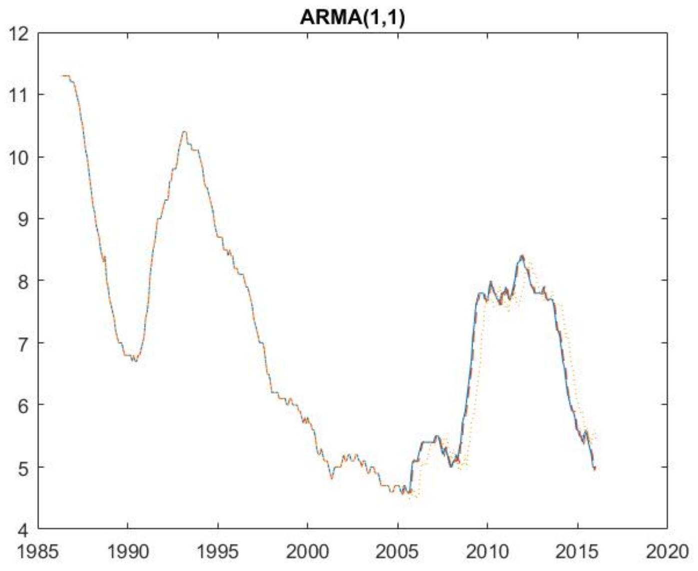

Figure 27: Forecast using ARMA(1,1)

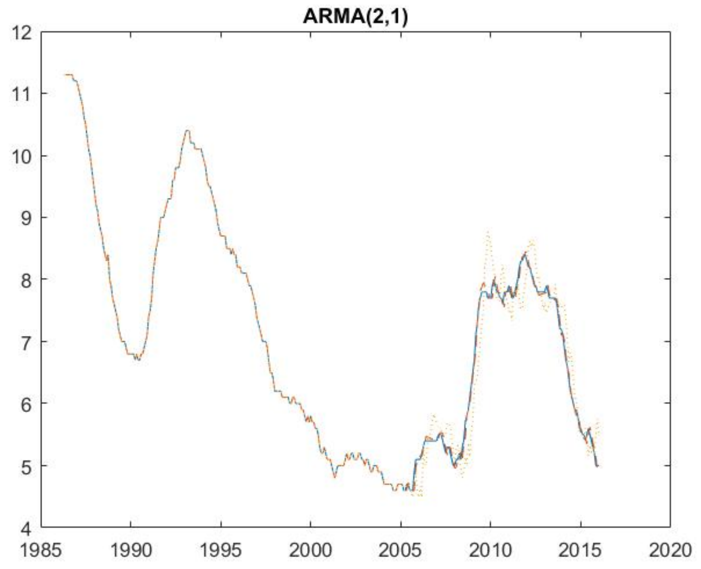

Figure 28: Forecast using ARMA(2,1)

2. Table 18 presents the mean-squared-forecast-error (MSFE) and mean-absolute-forecast-error (MAFE) for the 1 month ahead and the 6 months ahead forecast horizons.

Table 16: Forecast errors

	MSFE(1)	MAFE(1)	MSFE(6)	MAFE(6)
AR(2)	0.0095	0.0751	0.2244	0.3545
ARMA(1,1)	0.0110	0.0801	0.2591	0.3768
ARMA(2,1)	0.0088	0.0752	0.1805	0.3389

3. Based on the results we would prefer the ARMA(2,1) since it has the lowest forecast error for both horizons and both criteria (with the exception of the 1 step ahead for the MAFE where it is 0.0001 higher than the AR(2) model).

4 Forecast Evaluation

1. Table 17 presents the forecast errors for the 6 steps ahead forecast and the results of a Diebold-Mariano test of whether they are statistically significantly different. In both cases the Diebold-Mariano test rejects the null that the forecasts are equivalent.

Table 17: Forecast errors

	MSFE(6)	MAFE(6)
AR(2)	39858	111.3616
ARMA(1,1)	44.7673	6.5794
Diebold-Mariano	3.3124	3.5996
p-value	0.001	0.000

2. Based on these results (and with all aforementioned caveats) we would prefer the ARMA(1,1) model for forecasts 6 months ahead.

4.1 Using the built in function in Matlab

1. Table 18 presents the forecast errors for the 6 steps ahead forecast and the results of a Diebold-Mariano test of whether they are statistically significantly different. In both cases the Diebold-Mariano test rejects the null that the forecasts are equivalent.

Table 18: Diebold-Mariano test - MSFE

Against	AR(2)	ARMA(1,1)	ARMA(2,1)
AR(2)	-		
	-		
ARMA(1,1)	-1.6493	-	
	(0.0991)	-	
ARMA(2,1)	0.7343	1.0004	-
	(0.4628)	(0.3171)	-

Table 19: Diebold-Mariano test - MAFE

Against	AR(2)	ARMA(1,1)	ARMA(2,1)
AR(2)	-		
	-		
ARMA(1,1)	-1.6386	-	
	(0.1013)	-	
ARMA(2,1)	0.3484	0.7072	-
	(0.7276)	(0.4794)	-

2. Based on these results there is not a statistically significant difference in the forecast error variance between any of the models. So our model preference is not affected; we still prefer the ARMA(2,1) model.

5 Matlab code

Main file

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Adv. Econometric Methods III                                          %
%      Empirical Homework 1                                             %
%        - The objective of this problem set is forecasting             %
%          unemployment in the United Kingdom using ARMA models         %
%                                                                       %
% Team 3:                                                               %
% Suleman Dawood, Bjarni Einarsson, Adam Lee & Robertson Wang           %
%                                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Housekeeping
clear all
close all
```

```
%% Data prep
[data0, names] = xlsread('AllData.xlsx'); % load data
ur = data0(:,5);    % UK unemployment rate
year = data0(:,1);
month = data0(:,2);
T = size(data0,1);    % Number of periods in sample
TT = 1986.3333:(1/12):2016;
% Crisis start and end dates
startdate = [1973.25, 1974.5, 1979.25, 1990.25, 2008]';
enddate = [1974, 1975.5, 1981, 1991.5, 2009.25]';


%% Data Exploration
% 1. Plot of the time series
plot(TT',ur)
curax = axis;
title('UK unemployment rate')

% 2. Plot with recession indicator
figure
hold on
a4 = area([startdate(4) enddate(4)], [curax(4) curax(4)],curax(3));
a4.FaceColor = [0.8,0.8,0.8];
a4.EdgeColor = [0.8,0.8,0.8];
a5 = area([startdate(5) enddate(5)], [curax(4) curax(4)],curax(3));
a5.FaceColor = [0.8,0.8,0.8];
a5.EdgeColor = [0.8,0.8,0.8];
plot(TT',ur)
hold off
title('UK unemployment rate and recessions')

% 3. See .pdf

% 4. Autocorrelation function
figure
autocorr(ur)

% 5. Partial autocorrelation function
figure
parcorr(ur)

% 6. Histogram
figure
hist(ur)
title('Histogram of UK unemplyment rate')

%% 2 Models and Estimation

% 2.1 ARMA models
%    AR(2)
startvalAR2 = [0.1,0.1,0.1,0.1];
Aar2=[0,1,1,0;0,-1,1,0;0,0,1,0;0,0,-1,0];
```

```
bar2=[0.99999;0.99999;0.99999;0.99999];
[thetaAR2,fvalAR2,~,~,~,~,hessian] = fmincon(@(theta)lhoodAR2(theta,ur),startvalAR2,Aar2,bar2);
errAR2 =sqrt(diag(inv(hessian)));
resAR2 = zeros(T,1);
for i = 3:T
    resAR2(i) = ur(i)-thetaAR2(1) - thetaAR2(2)*ur(i-1) - thetaAR2(3)*ur(i-2);
end
figure
autocorr(resAR2)
figure
parcorr(resAR2)

%    ARMA(1,1)
startvalARMA11 = [0.1,0.1,0.1,0.1];
Aarma11=[0,1,0,0;0,-1,0,0];
barma11=[1;1];
[thetaARMA11,fvalARMA11,~,~,~,~,hessian] = fmincon(@(theta)lhoodARMA11(theta,ur),startvalARMA11,Aarma11
errARMA11=sqrt(diag(inv(hessian)));
resARMA11 = zeros(T,1);
for i = 2:T
    resARMA11(i) = ur(i)-thetaARMA11(1)-thetaARMA11(2)*ur(i-1)-thetaARMA11(3)*resARMA11(i-1);
end
figure
autocorr(resARMA11)
figure
parcorr(resARMA11)


% 2.2 Model selection
maxp = 2;
maxq = 2;
AIC = NaN(maxp+1,maxq+1);
BIC = NaN(maxp+1,maxq+1);
AICC = NaN(maxp+1,maxq+1);
loglik = NaN(maxp+1,maxq+1);
loglik(3,1) = fvalAR2;
loglik(2,2) = fvalARMA11;
% AR(1)
A=[0,1,0;0,-1,0];
b=[0.99999;0.99999];
theta0=[0.1,0.1,0.1];
[~,fvalAR1,~,~,~,~] = fmincon(@(theta)lhoodAR1(theta,ur),theta0,A,b);
loglik(2,1) = fvalAR1;
% MA(1)
theta0=[0.1,0.1,0.1];
[~,fvalMA1,~,~,~,~] = fminunc(@(theta)lhoodMA1(theta,ur),theta0);
loglik(1,2) = fvalMA1;
% MA(2)
theta0=[0.1,0.1,0.1,0.1];
[~,fvalMA2,~,~,~,~] = fminunc(@(theta)lhoodMA2(theta,ur),theta0);
loglik(1,3) = fvalMA2;
% ARMA(1,2)
theta0=[0.1,0.1,0.1,0.1,0.1];
A=[0,1,0,0,0;0,-1,0,0,0];
```

```
b=[1;1];
[~,fvalARMA12,~,~,~,~,~] = fmincon(@(theta)lhoodARMA12(theta,ur),theta0,A,b);
loglik(2,3) = fvalARMA12;
% ARMA(2,1)
A=[0,1,1,0,0;0,-1,1,0,0;0,0,1,0,0;0,0,-1,0,0];
b=[0.99999;0.99999;0.99999;0.99999];
theta0=[0.1,0.1,0.1,0.1,0.1];
[~,fvalARMA21,~,~,~,~,~] = fmincon(@(theta)lhoodARMA21(theta,ur),theta0,A,b);
loglik(3,2) = fvalARMA21;
% ARMA(2,2)
for p = 0:maxp
    for q = 0:maxq
        AIC(p+1,q+1) = 2*loglik(p+1,q+1) + 2*(p+q+1);
        BIC(p+1,q+1) = 2*loglik(p+1,q+1) + (p+q+1)*(log(T)/T);
        AICC(p+1,q+1) = 2*loglik(p+1,q+1) + 2*T*(p+q+1)/(T-p-q-2);
    end
end
% All criteria agree on AR(2) except BIC (ARMA(2,1)), AR(2) chosen for
% parsimony

%% 3 Forecasting
hh = 6; % longest forecast horizon
strtpt = 226; % Jan 2005 = obs 226
endpt = T;
yhatAR2 = zeros(endpt-strtpt,2);
yhatARMA = zeros(endpt-strtpt,2);
wait = waitbar(0,'MCMC algorithm is running...');
for i = 0:endpt-strtpt
    % Adjust sample
    smpl = ur(1:strtpt+i);
    % Estimate models
    [thetaAR2,~,~,~,~,~,~] = fmincon(@(theta)lhoodAR2(theta,smpl),startvalAR2,Aar2,bar2);
    [thetaARMA11,~,~,~,~,~,~] = fmincon(@(theta)lhoodARMA11(theta,smpl),startvalARMA11,Aarma11,barma11)
    resid = zeros(size(smpl,1),1);
    for j = 2:size(smpl,1)
        resid(j) = ur(j)-thetaARMA11(1)-thetaARMA11(2)*ur(j-1)-thetaARMA11(3)*resid(j-1);
    end
    % Calculate forecasts
    %   AR(2)
    temp = zeros(hh+2,1);
    temp(1:2) = smpl(end-1:end);
    for j = 3:length(temp)
        temp(j) = thetaAR2(1:end-1)*[1;flipud(temp(j-2:j-1))];
    end
    yhatAR2(i+1,:) = [temp(3) temp(end)];

    %   ARMA(1,1)
    temp = zeros(hh+1,2);
    temp(1,1) = smpl(end);
    temp(1,2) = resid(end);
    for j = 2:length(temp)
        temp(j) = thetaARMA11(1:end-1)*[1;temp(j-1,1);temp(j-1,2)];
    end
    yhatARMA(i+1,:) = [temp(2) temp(end)];
    % Waitbar progress
    if mod(i,5)==0
        waitbar(i/(endpt-strtpt), wait);
    end

end

% 1. Plot the forecasts
figure
plot(TT',ur, TT',[ur(1:strtpt);yhatAR2(1:end-1,1)],'--', TT',[ur(1:strtpt+hh-1);yhatAR2(1:end-6,2)],':'
title('AR(2)');

figure
plot(TT',ur, TT',[ur(1:strtpt);yhatARMA(1:end-1,1)],'--', TT',[ur(1:strtpt+hh-1);yhatARMA(1:end-6,2)],'
title('ARMA(1,1)')

% 2. MSFE & MAFE
% AR(2)
FE1sAR = ur(strtpt+1:end)-yhatAR2(1:end-1,1);
FE6sAR = ur(strtpt+hh:end)-yhatAR2(1:end-6,2);
MSFE1sAR = mean(FE1sAR.^2);
MSFE6sAR = mean(FE6sAR.^2);
MAFE1sAR = mean(abs(FE1sAR));
MAFE6sAR = mean(abs(FE6sAR));

% ARMA(1,1)
FE1sARMA = ur(strtpt+1:end)-yhatARMA(1:end-1,1);
FE6sARMA = ur(strtpt+hh:end)-yhatARMA(1:end-6,2);
MSFE1sARMA = mean(FE1sARMA.^2);
MSFE6sARMA = mean(FE6sARMA.^2);
MAFE1sARMA = mean(abs(FE1sARMA));
MAFE6sARMA = mean(abs(FE6sARMA));

% 3. See .pdf


%% 4 Forecast evaluation
%
[DMsq, pvalsq] = DieboldMariano(FE6sAR.^2,FE6sARMA.^2,6);
[DMabs, pvalabs] = DieboldMariano(abs(FE6sAR),abs(FE6sARMA),6);
```

Main file - using built in function in Matlab

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Adv. Econometric Methods III                                            %
%      Empirical Homework 1                                               %
%        - The objective of this problem set is forecasting               %
%          unemployment in the United Kingdom using ARMA models           %
%                                                                         %
% Team 3:                                                                 %
% Suleman Dawood, Bjarni Einarsson, Adam Lee & Robertson Wang             %
%                                                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%% Housekeeping
clear all
close all


%% Data prep
[data0, names] = xlsread('AllData.xlsx'); % load data
ur = data0(:,5);    % UK unemployment rate
year = data0(:,1);
month = data0(:,2);
T = size(data0,1);    % Number of periods in sample
TT = 1986.3333:(1/12):2016;
% Crisis start and end dates
startdate = [1973.25, 1974.5, 1979.25, 1990.25, 2008]';
enddate = [1974, 1975.5, 1981, 1991.5, 2009.25]';


%% Data Exploration
% 1. Plot of the time series
plot(TT',ur)
curax = axis;
title('UK unemployment rate')

% 2. Plot with recession indicator
figure
hold on
a4 = area([startdate(4) enddate(4)], [curax(4) curax(4)],curax(3));
a4.FaceColor = [0.8,0.8,0.8];
a4.EdgeColor = [0.8,0.8,0.8];
a5 = area([startdate(5) enddate(5)], [curax(4) curax(4)],curax(3));
a5.FaceColor = [0.8,0.8,0.8];
a5.EdgeColor = [0.8,0.8,0.8];
plot(TT',ur)
hold off
title('UK unemployment rate and recessions')

% 3. See .pdf

% 4. Autocorrelation function
figure
autocorr(ur)

% 5. Partial autocorrelation function
figure
parcorr(ur)

% 6. Histogram
figure
hist(ur)
title('Histogram of UK unemplyment rate')

%% 2 Models and Estimation
```

```
% 2.1 ARMA models
%    AR(2)
AR2 = arima(2,0,0);
[estAR2, covAR2, llAR2,~] = estimate(AR2,ur);
[resAR2,~] = infer(estAR2,ur);
figure
autocorr(resAR2)
figure
parcorr(resAR2)

%    ARMA(1,1)
ARMA11 = arima(1,0,1);
[estARMA11,covARMA11, llARMA11,~] = estimate(ARMA11,ur);
[resARMA11,~] = infer(estARMA11,ur);
figure
autocorr(resARMA11)
figure
parcorr(resARMA11)


% 2.2 Model selection
maxp = 5;
maxq = 5;
AIC = NaN(maxp+1,maxq+1);
BIC = NaN(maxp+1,maxq+1);
AICC = NaN(maxp+1,maxq+1);
loglik = NaN(maxp+1,maxq+1);
for q = 1:maxq
    tempmdl = arima(0,0,q);
    [~,~,llik,~] = estimate(tempmdl,ur);
    AIC(1,q+1) = -2*llik + 2*(q+1);
    BIC(1,q+1) = -2*llik + (q+1)*(log(T)/T);
    AICC(1,q+1) = -2*llik + 2*T*(q+1)/(T-q-2);
end
for p = 1:maxp
    for q = 0:maxq
        tempmdl = arima(p,0,q);
        [~,~,llik,~] = estimate(tempmdl,ur);
        AIC(p+1,q+1) = -2*llik + 2*(p+q+1);
        BIC(p+1,q+1) = -2*llik + (p+q+1)*(log(T)/T);
        AICC(p+1,q+1) = -2*llik + 2*T*(p+q+1)/(T-p-q-2);
    end
end

% Preferred model - ARMA(2,1)
ARMA21 = arima(2,0,1);
[estARMA21,covARMA21, llARMA21,~] = estimate(ARMA21,ur);


%% 3 Forecasting
hh = 6; % longest forecast horizon
strtpt = 226; % Jan 2005 = obs 226
endpt = T;
yhatAR2 = zeros(endpt-strtpt,2);
```

```
yhatARMA11 = zeros(endpt-strtpt,2);
yhatARMA21 = zeros(endpt-strtpt,2);
wait = waitbar(0,'Forecasting algorithm is running...');
for i = 0:endpt-strtpt
    % Adjust sample
    smpl = ur(1:strtpt+i);
    % Estimate models
    %  AR(2)
    mdlAR2 =arima(2,0,0);
    estmdlAR2 = estimate(mdlAR2,smpl);

    %  ARMA(1,1)
    mdlARMA11 = arima(1,0,1);
    estmdlARMA11 = estimate(mdlARMA11,smpl);
    [resARMA11,~]=infer(estmdlARMA11,smpl);

    %  ARMA(2,1)
    mdlARMA21 = arima(2,0,1);
    estmdlARMA21 = estimate(mdlARMA21,smpl);
    [resARMA21,~] = infer(estmdlARMA21,smpl);

    % Calculate forecasts
    %   AR(2)
    temp = zeros(hh+2,1);
    temp(1:2) = smpl(end-1:end);
    for j = 3:length(temp)
        temp(j) = [estmdlAR2.Constant cell2mat(estmdlAR2.AR)]*[1;flipud(temp(j-2:j-1))];
    end
    yhatAR2(i+1,:) = [temp(3) temp(end)];

    %   ARMA(1,1)
    temp = zeros(hh+1,2);
    temp(1,1) = smpl(end);
    temp(1,2) = resARMA11(end);
    for j = 2:length(temp)
        temp(j) = [estmdlARMA11.Constant cell2mat(estmdlARMA11.AR) cell2mat(estmdlARMA11.MA)]*[1;temp(j
    end
    yhatARMA11(i+1,:) = [temp(2,1) temp(end,1)];

    %   ARMA(2,1)
    temp = zeros(hh+2,2);
    temp(1:2,1) = smpl(end-1:end);
    temp(2,2) = resARMA21(end);
    for j = 3:length(temp)
        temp(j) = [estmdlARMA21.Constant cell2mat(estmdlARMA21.AR) cell2mat(estmdlARMA21.MA)]*[1;flipud
    end
    yhatARMA21(i+1,:) = [temp(3,1) temp(end,1)];

    % Waitbar progress
    if mod(i,5)==0
        waitbar(i/(endpt-strtpt), wait);
    end

end
```

```
close(wait);

% 1. Plot the forecasts
figure
plot(TT',ur, TT',[ur(1:strtpt);yhatAR2(1:end-1,1)],'--', TT',[ur(1:strtpt+hh-1);yhatAR2(1:end-6,2)],':'
title('AR(2)');

figure
plot(TT',ur, TT',[ur(1:strtpt);yhatARMA11(1:end-1,1)],'--', TT',[ur(1:strtpt+hh-1);yhatARMA11(1:end-6,2
title('ARMA(1,1)')

figure
plot(TT',ur, TT',[ur(1:strtpt);yhatARMA21(1:end-1,1)],'--', TT',[ur(1:strtpt+hh-1);yhatARMA21(1:end-6,2
title('ARMA(2,1)')

% 2. MSFE & MAFE
% AR(2)
FE1sAR = ur(strtpt+1:end)-yhatAR2(1:end-1,1);
FE6sAR = ur(strtpt+hh:end)-yhatAR2(1:end-6,2);
MSFE1sAR = mean(FE1sAR.^2);
MSFE6sAR = mean(FE6sAR.^2);
MAFE1sAR = mean(abs(FE1sAR));
MAFE6sAR = mean(abs(FE6sAR));

% ARMA(1,1)
FE1sARMA11 = ur(strtpt+1:end)-yhatARMA11(1:end-1,1);
FE6sARMA11 = ur(strtpt+hh:end)-yhatARMA11(1:end-6,2);
MSFE1sARMA11 = mean(FE1sARMA11.^2);
MSFE6sARMA11 = mean(FE6sARMA11.^2);
MAFE1sARMA11 = mean(abs(FE1sARMA11));
MAFE6sARMA11 = mean(abs(FE6sARMA11));

% ARMA(2,1)
FE1sARMA21 = ur(strtpt+1:end)-yhatARMA21(1:end-1,1);
FE6sARMA21 = ur(strtpt+hh:end)-yhatARMA21(1:end-6,2);
MSFE1sARMA21 = mean(FE1sARMA21.^2);
MSFE6sARMA21 = mean(FE6sARMA21.^2);
MAFE1sARMA21 = mean(abs(FE1sARMA21));
MAFE6sARMA21 = mean(abs(FE6sARMA21));

% 3. See .pdf


%% 4 Forecast evaluation
% AR2 vs ARMA11
[DMsq1, pvalsq1] = DieboldMariano(FE6sAR.^2,FE6sARMA11.^2,6);
[DMabs1, pvalabs1] = DieboldMariano(abs(FE6sAR),abs(FE6sARMA11),6);

% AR2 vs ARMA21
[DMsq2, pvalsq2] = DieboldMariano(FE6sAR.^2,FE6sARMA21.^2,6);
[DMabs2, pvalabs2] = DieboldMariano(abs(FE6sAR),abs(FE6sARMA21),6);

% ARMA11 vs ARMA21
[DMsq3, pvalsq3] = DieboldMariano(FE6sARMA11.^2,FE6sARMA21.^2,6);
```

```
[DMabs3, pvalabs3] = DieboldMariano(abs(FE6sARMA11),abs(FE6sARMA21),6);
```

User-defined functions used above

```
%% Implements the Asymptotic test in Sec 1.1. of Diebold Mariano (1995)
%
% INPUTS
%   FE1: Forecast error 1
%   FE2: Forecast error 2
%   steps: how many steps forward the forecast is
%
% OUTPUTS
%   pval: the p-value of the test of the null of no difference
%
function [DM, pval] = DieboldMariano(FE1,FE2,steps)

%%
T = size(FE1,1);  % number of observations
d = FE1-FE2;    % forecast error difference
dbar = mean(d);
gamma0 = var(d);
if steps > 1    % correction for autocovariance if more than 1 step forecast
    gamma = zeros(steps-1,1);
    for i = 1:steps-1
        covbar = cov(d(1+i:T),d(1:T-i));
        gamma(i) = covbar(2);
    end
    dvar = gamma0 + 2*sum(gamma);
else
    dvar = gamma0;
end
DM = dbar /sqrt((1/T)*dvar);        % Test statistic
pval = 2*(1-cdf('Normal',abs(DM),0,1));  % p-value


function ACVF=ACVF_AR1(h,Phi,sigmasq)
% This computes the ACVF at lag h for a AR(1) model and returns a
% vector of all lags up to h.

if ~(length(Phi)==1)
    error('Length of coefficient vector is not 1')
else

  ACVF_0=(sigmasq)/(1-Phi^2);
  ACVF=zeros(h+1,1);
  ACVF(1)=[ACVF_0];
     if h>0
          for i=2:h+1
              ACVF(i)=Phi^(i-1)*ACVF_0;
          end
      end
  ACVF=ACVF(1:h+1);
```

```
end


function ACVF=ACVF_AR2(h,Phi,sigmasq)
% This computes the ACVF at lag h for a  AR(2) model and returns a
% vector of all lags up to h.

if ~(length(Phi)==2)
    error('Length of coefficient vector is not 2')
else
  ACVF_0=(sigmasq)/(1-((Phi(1))^2/(1-Phi(2)))-((Phi(2)*Phi(1)^2)/(1-Phi(2)))-Phi(2)^2);
  ACVF_1=(Phi(1)/(1-Phi(2)))*ACVF_0;
  ACVF_2=Phi(1)*ACVF_1+Phi(2)*ACVF_1;
  ACVF=zeros(h+1,1);
  ACVF(1:3)=[ACVF_0;ACVF_1;ACVF_2];
      if h>2
          for i=4:h+1
              ACVF(i)=Phi(1)*ACVF(i-1)+Phi(2)*ACVF(i-2);
          end
      end
  ACVF=ACVF(1:h+1);
end


function ACVF=ACVF_MAq(q,h,Theta,sigmasq)
% This computes the ACVF at lag h for a  MA(q) model and returns a
% vector of all lags up to h.

if ~(length(Theta)==q)
    error('Length of coefficient vector is not correct')
else



       s=1;
    for j=1:q
        s=s+Theta(j)^2;
    end
    ACVF_0=sigmasq*s;
    ACVF=zeros(h+1,1);
    ACVF(1)=ACVF_0;
          for i=2:h+1
              if (i-1)<=q
                  sum=Theta(i-1);
                  for k=i:q
                      sum=sum+Theta(k)*Theta(k-i+1);
                  end
              ACVF(i)=sigmasq*sum;
              else
              ACVF(i)=0;
              end
          end
      end
  ACVF=ACVF(1:h+1);
end
```

```
function ACVF=ACVF_ARMA11(h,Phi,Theta,sigmasq)
% This computes the ACVF at lag h for a ARMA(1,1) model and returns a
% vector of all lags up to h.

if ~(length(Phi)==1)
    error('Length of coefficient vector is not 1')
end
if ~(length(Theta)==1)
    error('Length of coefficient vector is not 1')
end


  ACVF_0=(sigmasq)*(1+((Theta+Phi)^2)/(1-Phi^2));
  ACVF_1=(sigmasq)*(Theta+Phi+(((Theta+Phi)^2)*Phi)/(1-Phi^2));
  ACVF=zeros(h+1,1);
  ACVF(1:2)=[ACVF_0;ACVF_1];
      if h>1
          for i=3:h+1
              ACVF(i)=(Phi^(i-2))*ACVF_1;
          end
      end
  ACVF=ACVF(1:h+1);
end


function ACVF=ACVF_ARMA21(h,Phi,Theta,sigmasq)
% This computes the ACVF at lag h for a ARMA(2,1) model and returns a
% vector of all lags up to h.

if ~(length(Phi)==2)
    error('Length of coefficient vector is not 2')
end
if ~(length(Theta)==1)
    error('Length of coefficient vector is not 1')
end

 ACVF_0=(1/(1-((Phi(1)^2)/(1-Phi(2)))-(Phi(2)*Phi(1)^2)/(1-Phi(2))-Phi(2)^2))*(sigmasq*(1+Theta*Phi(1)+
 ACVF_1=(Phi(1)*ACVF_0+Theta*sigmasq)/(1-Phi(2));
 ACVF=zeros(h+1,1);
 ACVF(1:2)=[ACVF_0;ACVF_1];
      if h>1
          for i=3:h+1
              ACVF(i)=Phi(1)*ACVF(i-1)+Phi(2)*ACVF(i-2);
          end
      end
  ACVF=ACVF(1:h+1);
end


function ACVF=ACVF_ARMA12(h,Phi,Theta,sigmasq)
% This computes the ACVF at lag h for a ARMA(1,2) model and returns a
% vector of all lags up to h.

if ~(length(Phi)==1)
    error('Length of coefficient vector is not 1')
```

```
end
if ~(length(Theta)==2)
    error('Length of coefficient vector is not 2')
end


 ACVF_0=(sigmasq/(1-Phi^2))*(1+Theta(1)^2+Theta(1)*Phi+Theta(2)^2+Phi*Theta(1)*Theta(2)+Theta(1)+Theta(
 ACVF_1=Phi*ACVF_0+sigmasq*(Theta(1)+Theta(2)*Theta(1)+Theta(2));
 ACVF_2=Phi*ACVF_1+Theta(2)*sigmasq;
  ACVF=zeros(h+1,1);
  ACVF(1:3)=[ACVF_0;ACVF_1;ACVF_2];
      if h>2
          for i=4:h+1
              ACVF(i)=Phi*ACVF(i-1);
          end
      end
  ACVF=ACVF(1:h+1);
end


%% Likelihood function calculated via Durbin Levinson
function [l] = lhoodAR1(paramvector,y)
% Paramvector is the vector of parameters
c=paramvector(1);
Phi=paramvector(2);
sigmasq=paramvector(3);

% y is the timeseries - sample data
n=length(y);
%calculate ACVF
A=ACVF_AR1(n-1,Phi,sigmasq);
% Starting values for D-L recursion & lhood
v=A(1);
l=-0.5*(log(v)+((y(1)-c)^2)/v);

a=A(2)/A(1);
v=A(1)*(1-a^2);
p=c+a*y(1);
l=l-0.5*(log(v)+((y(2)-p)^2)/v);
% D-L recursion & calculation of lhood
for i=2:n-1
     a_n=(A(i+1)-a'*A(i:-1:2))/v;
    a_1=a-a_n*[flipud(a)];
    a=[a_1;a_n];
    v=v*(1-(a_n)^2);
    p=c+a'*y(i:-1:1);
    l=l-0.5*(log(v)+((y(i+1)-p)^2)/v);
end

l=-(l-n/(2*pi));
end


%% Likelihood function calculated via Durbin Levinson
function [l] = lhoodAR2(paramvector,y)
```

```
% Paramvector is the vector of parameters
c=paramvector(1);
Phi=paramvector(2:3);
sigmasq=paramvector(4);

% y is the timeseries - sample data
n=length(y);
%calculate ACVF
A=ACVF_AR2(n-1,Phi,sigmasq);
% Starting values for D-L recursion & lhood
v=A(1);
l=-0.5*(log(v)+((y(1)-c)^2)/v);

a=A(2)/A(1);
v=A(1)*(1-a^2);
p=c+a*y(1);
l=l-0.5*(log(v)+((y(2)-p)^2)/v);
% D-L recursion & calculation of lhood
for   i=2:n-1
    a_n=(A(i+1)-a'*A(i:-1:2))/v;
    a_1=a-a_n*[flipud(a)];
    a=[a_1;a_n];
    v=v*(1-(a_n)^2);
    p=c+a'*y(i:-1:1);
    l=l-0.5*(log(v)+((y(i+1)-p)^2)/v);
end

l=-(l-n/(2*pi));
end


%% Likelihood function calculated via Durbin Levinson
function [l] = lhoodARMA11(paramvector,y)
% Paramvector is the vector of parameters
c=paramvector(1);
Phi=paramvector(2);
Theta=paramvector(3);
sigmasq=paramvector(4);

% y is the timeseries - sample data
n=length(y);
%calculate ACVF
A=ACVF_ARMA11(n-1,Phi,Theta,sigmasq);
% Starting values for D-L recursion & lhood
v=A(1);
l=-0.5*(log(v)+((y(1)-c)^2)/v);

a=A(2)/A(1);
v=A(1)*(1-a^2);
p=c+a*y(1);
l=l-0.5*(log(v)+((y(2)-p)^2)/v);
% D-L recursion & calculation of lhood
for   i=2:n-1
    a_n=(A(i+1)-a'*A(i:-1:2))/v;
    a_1=a-a_n*[flipud(a)];
    a=[a_1;a_n];
    v=v*(1-(a_n)^2);
    p=c+a'*y(i:-1:1);
    l=l-0.5*(log(v)+((y(i+1)-p)^2)/v);
end

l=-(l-n/(2*pi));
end


%% Likelihood function calculated via Durbin Levinson
function [l] = lhoodARMA12(paramvector,y)
% Paramvector is the vector of parameters
c=paramvector(1);
Phi=paramvector(2);
Theta=paramvector(3:4);
sigmasq=paramvector(5);

% y is the timeseries - sample data
n=length(y);
%calculate ACVF
A=ACVF_ARMA12(n-1,Phi,Theta,sigmasq);
% Starting values for D-L recursion & lhood
v=A(1);
l=-0.5*(log(v)+((y(1)-c)^2)/v);

a=A(2)/A(1);
v=A(1)*(1-a^2);
p=c+a*y(1);
l=l-0.5*(log(v)+((y(2)-p)^2)/v);
% D-L recursion & calculation of lhood
for   i=2:n-1
    a_n=(A(i+1)-a'*A(i:-1:2))/v;
    a_1=a-a_n*[flipud(a)];
    a=[a_1;a_n];
    v=v*(1-(a_n)^2);
    p=c+a'*y(i:-1:1);
    l=l-0.5*(log(v)+((y(i+1)-p)^2)/v);
end

l=-(l-n/(2*pi));
end


%% Likelihood function calculated via Durbin Levinson
function [l] = lhoodARMA21(paramvector,y)
% Paramvector is the vector of parameters
c=paramvector(1);
Phi=paramvector(2:3);
Theta=paramvector(4);
sigmasq=paramvector(5);

% y is the timeseries - sample data
n=length(y);
%calculate ACVF
```

```
A=ACVF_ARMA21(n-1,Phi,Theta,sigmasq);
% Starting values for D-L recursion & lhood
v=A(1);
l=-0.5*(log(v)+((y(1)-c)^2)/v);

a=A(2)/A(1);
v=A(1)*(1-a^2);
p=c+a*y(1);
l=l-0.5*(log(v)+((y(2)-p)^2)/v);
% D-L recursion & calculation of lhood
for   i=2:n-1
    a_n=(A(i+1)-a'*A(i:-1:2))/v;
    a_1=a-a_n*[flipud(a)];
    a=[a_1;a_n];
    v=v*(1-(a_n)^2);
    p=c+a'*y(i:-1:1);
    l=l-0.5*(log(v)+((y(i+1)-p)^2)/v);
end

l=-(l-n/(2*pi));
end


%% Likelihood function calculated via Durbin Levinson
function [l] = lhoodMA1(paramvector,y)
% Paramvector is the vector of parameters
c=paramvector(1);
Theta=paramvector(2);
sigmasq=paramvector(3);

% y is the timeseries - sample data
n=length(y);
%calculate ACVF
A=ACVF_MAq(1,n-1,Theta,sigmasq);
% Starting values for D-L recursion & lhood
v=A(1);
l=-0.5*(log(v)+((y(1)-c)^2)/v);

a=A(2)/A(1);
v=A(1)*(1-a^2);
p=c+a*y(1);
l=l-0.5*(log(v)+((y(2)-p)^2)/v);
% D-L recursion & calculation of lhood
for i=2:n-1
     a_n=(A(i+1)-a'*A(i:-1:2))/v;
    a_1=a-a_n*[flipud(a)];
    a=[a_1;a_n];
    v=v*(1-(a_n)^2);
    p=c+a'*y(i:-1:1);
    l=l-0.5*(log(v)+((y(i+1)-p)^2)/v);
end

l=-(l-n/(2*pi));
end
```

```
%% Likelihood function calculated via Durbin Levinson
function [l] = lhoodMA2(paramvector,y)
% Paramvector is the vector of parameters
c=paramvector(1);
Theta=paramvector(2:3);
sigmasq=paramvector(4);

% y is the timeseries - sample data
n=length(y);
%calculate ACVF
A=ACVF_MAq(2,n-1,Theta,sigmasq);
% Starting values for D-L recursion & lhood
v=A(1);
l=-0.5*(log(v)+((y(1)-c)^2)/v);

a=A(2)/A(1);
v=A(1)*(1-a^2);
p=c+a*y(1);
l=l-0.5*(log(v)+((y(2)-p)^2)/v);
% D-L recursion & calculation of lhood
for i=2:n-1
     a_n=(A(i+1)-a'*A(i:-1:2))/v;
    a_1=a-a_n*[flipud(a)];
    a=[a_1;a_n];
    v=v*(1-(a_n)^2);
    p=c+a'*y(i:-1:1);
    l=l-0.5*(log(v)+((y(i+1)-p)^2)/v);
end

l=-(l-n/(2*pi));
end
```

Monte Carlo Simulation

```
%% Monte Carlo Simulations of the AR(2) and ARMA(1,1) estimation

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This .m file performs various monte carlo simulations of our
% routines for estimating AR(2) and ARMA(1,1) models.


%% AR(2) Test 1

n=100;

resultsAR2=zeros(n,4);

for j=1:n

    Mdl = arima('AR',{0.8,0.15},'Constant',0,'Variance',0.04);
    y=simulate(Mdl,200);

    A=[0,1,1,0;0,-1,1,0;0,0,1,0;0,0,-1,0];
    b=[0.99999;0.99999;0.99999;0.99999];
    theta0=[0.1,0.1,0.1,0.1];
    theta= fmincon(@(theta)lhoodAR2(theta,y),theta0,A,b);

    resultsAR2(j,:)=theta;
    j
end
mean(resultsAR2)
median(resultsAR2)
plot(resultsAR2)


%% AR(2) Test 2

n=200;

resultsAR2=zeros(n,4);

for j=1:n

    Mdl = arima('AR',{0.6,0.2},'Constant',0,'Variance',(0.2)^2);
    y=simulate(Mdl,200);

    A=[0,1,1,0;0,-1,1,0;0,0,1,0;0,0,-1,0];
    b=[0.99999;0.99999;0.99999;0.99999];
    theta0=[0.1,0.1,0.1,0.1];
    theta= fmincon(@(theta)lhoodAR2(theta,y),theta0,A,b);

    resultsAR2(j,:)=theta;
    j
end
mean(resultsAR2)
median(resultsAR2)
plot(resultsAR2)


%% AR(2) Test 3

n=100;

resultsAR2=zeros(n,4);

for j=1:n

    Mdl = arima('AR',{0.95,0.03},'Constant',0,'Variance',(0.2)^2);
    y=simulate(Mdl,200);

    A=[0,1,1,0;0,-1,1,0;0,0,1,0;0,0,-1,0];
    b=[0.99999;0.99999;0.99999;0.99999];
    theta0=[0.1,0.1,0.1,0.1];
    theta= fmincon(@(theta)lhoodAR2(theta,y),theta0,A,b);

    resultsAR2(j,:)=theta;
    j
end
```

```
mean(resultsAR2)
median(resultsAR2)
plot(resultsAR2)


%% AR(2) Test 4

n=100;

resultsAR2=zeros(n,4);

for j=1:n

    Mdl = arima('AR',{0.99,0.005},'Constant',0,'Variance',(0.2)^2);
    y=simulate(Mdl,200);

    A=[0,1,1,0;0,-1,1,0;0,0,1,0;0,0,-1,0];
    b=[0.99999;0.99999;0.99999;0.99999];
    theta0=[0.1,0.1,0.1,0.1];
    theta= fmincon(@(theta)lhoodAR2(theta,y),theta0,A,b);

    resultsAR2(j,:)=theta;
    j
end
mean(resultsAR2)
median(resultsAR2)
plot(resultsAR2)


%% ARMA(1,1) test 1

n=100;
resultsARMA11=zeros(n,4);

for k=1:n

    Mdl = arima('AR',{0.6},'D',0,'MA',{0.2},'Constant',0,'Variance',0.04);
    x=simulate(Mdl,200);

    A=[0,1,0,0;0,-1,0,0];
    b=[0.99999;0.99999];
    theta0=[0.1,0.1,0.1,0.1];
    theta=fmincon(@(theta)lhoodARMA11(theta,x),theta0,A,b);

    resultsARMA11(k,:)=theta;
    k
end

mean(resultsARMA11)
median(resultsARMA11)
plot(resultsARMA11)

%% ARMA(1,1) test 2
```

```
n=100;
resultsARMA11=zeros(n,4);

for k=1:n

    Mdl = arima('AR',{0.8},'D',0,'MA',{0.15},'Constant',0,'Variance',0.04);
    x=simulate(Mdl,200);

    A=[0,1,0,0;0,-1,0,0];
    b=[0.99999;0.99999];
    theta0=[0.1,0.1,0.1,0.1];
    theta=fmincon(@(theta)lhoodARMA11(theta,x),theta0,A,b);

    resultsARMA11(k,:)=theta;
    k
end

mean(resultsARMA11)
median(resultsARMA11)
plot(resultsARMA11)


%% ARMA(1,1) test 3

n=100;
resultsARMA11=zeros(n,4);

for k=1:n

    Mdl = arima('AR',{0.95},'D',0,'MA',{0.03},'Constant',0,'Variance',0.04);
    x=simulate(Mdl,200);

    A=[0,1,0,0;0,-1,0,0];
    b=[0.99999;0.99999];
    theta0=[0.1,0.1,0.1,0.1];
    theta=fmincon(@(theta)lhoodARMA11(theta,x),theta0,A,b);

    resultsARMA11(k,:)=theta;
    k
end

mean(resultsARMA11)
median(resultsARMA11)
plot(resultsARMA11)


%% ARMA(1,1) test 4

n=100;
resultsARMA11=zeros(n,4);

for k=1:n

    Mdl = arima('AR',{0.99},'D',0,'MA',{0.005},'Constant',0,'Variance',0.04);
    x=simulate(Mdl,200);

    A=[0,1,0,0;0,-1,0,0];
    b=[0.99999;0.99999];
    theta0=[0.1,0.1,0.1,0.1];
    theta=fmincon(@(theta)lhoodARMA11(theta,x),theta0,A,b);

    resultsARMA11(k,:)=theta;
    k
end

mean(resultsARMA11)
median(resultsARMA11)
plot(resultsARMA11)
```