Problem Set 3

Econ 211C

Question 1 .... *70 points*

Recall the $ARMA(2,5)$ process in Problem Set 1:

$$Y_t = 1.3Y_{t-1} - 0.4Y_{t-2} + \varepsilon_t + 0.7\varepsilon_{t-1} + 0.1\varepsilon_{t-3} - 0.5\varepsilon_{t-4} - 0.2\varepsilon_{t-5},$$

where $\varepsilon_t \sim WN(0,1)$.

(a) (5 points) What are the exact, finite-sample, one-step forecast coefficients?

Solution: In the Problem Set 1 we solved for the variance and first five autocovariances, which are reported in the table below.

γ_0	γ_1	γ_2	γ_3	γ_4	γ_5
17.5170	15.9570	12.4010	8.3985	5.0576	3.0155

To compute the exact finite-sample forecast coefficients with $m = 5$ observations, we solve:

$$\boldsymbol{\beta}^{(5,1)} = \begin{bmatrix} \gamma_0 & \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 \\ \gamma_1 & \gamma_0 & \gamma_1 & \gamma_2 & \gamma_3 \\ \gamma_2 & \gamma_1 & \gamma_0 & \gamma_1 & \gamma_2 \\ \gamma_3 & \gamma_2 & \gamma_1 & \gamma_0 & \gamma_1 \\ \gamma_4 & \gamma_3 & \gamma_2 & \gamma_1 & \gamma_0 \end{bmatrix}^{-1} \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \gamma_4 \\ \gamma_5 \end{bmatrix}.$$

The following `R` code finds the coefficients to be the values reported in the table below.

$\beta_0^{(5,1)}$	$\beta_1^{(5,1)}$	$\beta_2^{(5,1)}$	$\beta_3^{(5,1)}$	$\beta_4^{(5,1)}$
1.771	-1.206	0.4434	-0.2490	0.1519

```
# Autocovariances
g0 = 17.5170
g1 = 15.9570
g2 = 12.4010
g3 = 8.3985
g4 = 5.0576
g5 = 3.0155
```

```
# Comput the one-step coefficients
gamMat = matrix(c(g0,g1,g2,g3,g4,
  g1,g0,g1,g2,g3,
  g2,g1,g0,g1,g2,
  g3,g2,g1,g0,g1,
  g4,g3,g2,g1,g0), ncol=5, byrow=TRUE)
gamVec = c(g1,g2,g3,g4,g5)
beta51 = solve(gamMat)%*%gamVec
```

(b) (8 points) Simulate $N = 105$ observations for this process. Starting with Y_{100}, compute and report one-step forecasts for $Y_{101}, \ldots, Y_{105}$. When computing the forecasts for $t \geq 102$, use your previously computed forecasts as data, rather than the actual values that you originally simulated.

Solution: The R code below produced the forecasts in the following table.

$\hat{Y}_{101}$	$\hat{Y}_{102}$	$\hat{Y}_{103}$	$\hat{Y}_{104}$	$\hat{Y}_{105}$
-1.343	-2.045	-2.967	-3.961	-4.730

```
# Simulate data
N = 105
arCoef = c(1.3,-0.4)
maCoef = c(0.7,0,0.1,-0.5,-0.2)
Y = arima.sim(model=list(ar=arCoef,ma=maCoef),N)

# Compute 5, one-step forecasts
yHat = rev(Y[1:100])
for(i in 1:5){
yHat = c(yHat[1:5]%*%beta51,yHat)
}
```

(c) (8 points) Repeat part (b) 1000 times. What is the mean squared error of your forecast for Y_{105}?

Solution: Using the following code, I found the MSE to be 17.10.

```
nSim = 1000
```

```
sqErrors = rep(0,nSim)
for(j in 1:nSim){
  Y = arima.sim(model=list(ar=arCoef,ma=maCoef),N)
  yHat = rev(Y[1:100])
  for(i in 1:5){
    yHat = c(yHat[1:5]%*%beta51,yHat)
  }
  sqErrors[j] = (yHat[1] - rev(Y)[1])^2
}
mean(sqErrors)
```

(d) (5 points) What are the exact, finite-sample, five-step forecast coefficients?

Solution: From Problem Set 1, we know that $\gamma_j = 1.3\gamma_{j-1} - 0.4\gamma_{j-2}$ for $j \geq 6$. Using this, we compute the exact finite-sample forecast coefficients with $m = 5$ observations by solving:

$$\boldsymbol{\beta}^{(5)} = \begin{bmatrix} \gamma_0 & \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 \\ \gamma_1 & \gamma_0 & \gamma_1 & \gamma_2 & \gamma_3 \\ \gamma_2 & \gamma_1 & \gamma_0 & \gamma_1 & \gamma_2 \\ \gamma_3 & \gamma_2 & \gamma_1 & \gamma_0 & \gamma_1 \\ \gamma_4 & \gamma_3 & \gamma_2 & \gamma_1 & \gamma_0 \end{bmatrix}^{-1} \begin{bmatrix} \gamma_5 \\ \gamma_6 \\ \gamma_7 \\ \gamma_8 \\ \gamma_9 \end{bmatrix}.$$

The following `R` code finds the coefficients to be the values reported in the table below.

$\beta_0^{(5,5)}$	$\beta_1^{(5,5)}$	$\beta_2^{(5,5)}$	$\beta_3^{(5,5)}$	$\beta_4^{(5,5)}$
0.6606	-0.7879	0.3226	0.0108	-0.0147

```
g6 = 1.3*g5 - 0.4*g4
g7 = 1.3*g6 - 0.4*g5
g8 = 1.3*g7 - 0.4*g6
g9 = 1.3*g8 - 0.4*g7
gamVec5 = c(g5,g6,g7,g8,g9)
beta55 = solve(gamMat)%*%gamVec5
```

(e) (6 points) Compute and report a five-step forecast for Y_{105}.

Solution: The single forecast is 0.07663, and easily computed by the R command:

```
yHat105 = rev(Y[96:100])%*%beta55
```

(f) (6 points) Repeat part (e) 1000 times. What is the mean squared error of your forecast for Y_{105}?

Solution: Using the following code, I found the MSE to be 16.62.

```
# Repeat 1000 times
sqErrors2 = rep(0,nSim)
for(j in 1:nSim){
yHat105 = rev(yDatMat[96:100,j])%*%beta55
sqErrors2[j] = (yHat105 - rev(yDatMat[,j])[1])^2
}
MSE2 = mean(sqErrors2)
```

For the remainder of the problem, assume that you do not know the true coefficients of the process, but that you do know that it is an $ARMA(2,5)$.

(g) (5 points) Use the first 100 observations, $\{y_t\}_{t=1}^{100}$ to estimate the $ARMA(2,5)$. What are the parameter estimates?

Solution: Using the same data that was simulated in part (b), the $ARMA(2,5)$ estimates (with no intercept) are reported below, with associated R code.

$\hat{\phi}_1$	$\hat{\phi}_2$	$\hat{\theta}_1$	$\hat{\theta}_2$	$\hat{\theta}_3$	$\hat{\theta}_4$	$\hat{\theta}_5$	$\hat{\sigma}^2$
1.6487	-0.6685	0.2478	-0.3764	-0.0640	-0.5359	-0.2716	0.9499

```
# Estimate the ARMA(2,5)
arima(yDat,order=c(2,0,5),include.mean=FALSE)
```

(h) (5 points) Repeat part (a), using your estimates.

Solution: To start, we need to compute estimates of the $ARMA(2,5)$ autocovariances. We can do this by repeating the tedious work (by hand) in Problem Set 1, using the new, estimated, coefficients. Alternatively, we can cheat by using the `ARMAacf` function in R, which computes the exact $ARMA$ autocovariances for a specified model. My autocovariance estimates and forecast coefficients are reported below, along with R code.

$\hat{\gamma}_0$	$\hat{\gamma}_1$	$\hat{\gamma}_2$	$\hat{\gamma}_3$	$\hat{\gamma}_4$	$\hat{\gamma}_5$
16.1964	15.1740	13.0199	10.7071	8.6861	7.3487

$\beta_0^{(5,1)}$	$\beta_1^{(5,1)}$	$\beta_2^{(5,1)}$	$\beta_3^{(5,1)}$	$\beta_4^{(5,1)}$
1.670	-1.089	0.5721	-0.4728	0.2613

```
# Estimate exact autocorrelations and forecast coefficients
gamVecEst = ARMAacf(ar=armaEst$coef[1:2],ma=armaEst$coef[3:7])[2:6]*var(yDat)
g0Est = var(yDat)
g1Est = gamVecEst[1]
g2Est = gamVecEst[2]
g3Est = gamVecEst[3]
g4Est = gamVecEst[4]
gamMatEst = matrix(c(g0Est,g1Est,g2Est,g3Est,g4Est,
  g1Est,g0Est,g1Est,g2Est,g3Est,
  g2Est,g1Est,g0Est,g1Est,g2Est,
  g3Est,g2Est,g1Est,g0Est,g1Est,
  g4Est,g3Est,g2Est,g1Est,g0Est), ncol=5, byrow=TRUE)
beta51Est = solve(gamMatEst)%*%gamVecEst
```

(i) (5 points) Repeat parts (b) and (c), using the same estimates (without updating) for each one-step forecast.

Solution: Repeating the forecasting preceedure 10,000 times with the forecast coefficients obtained from the estimated model, I found the MSE to be 18.75. The `R` code is below.

```
# Compute 5 one-step MSE for estimated model, using 10000 reps
nSim = 10000
sqErrors1 = rep(0,nSim)
yDatMat = matrix(0,nrow=N,ncol=nSim)
for(j in 1:nSim){
yDatMat[,j] = arima.sim(model=list(ar=armaEst$coef[1:2],ma=armaEst$coef[3:7]),N)
yHat = rev(yDatMat[1:100,j])
for(i in 1:5){
```

```
yHat = c(yHat[1:5]%*%beta51Est,yHat)
}
sqErrors1[j] = (yHat[1] - rev(yDatMat[,j])[1])^2
}
MSE1Est = mean(sqErrors1)
```

(j) (7 points) Repeat parts (b) and (c), updating your estimates with each forecast. That is, compute the forecast for Y_{101}, using estimates obtained from $\{y_t\}_{t=1}^{100}$, compute the forecast for Y_{102} using estimates obtained from $\{y_t\}_{t=2}^{100}$ and your forecast $\hat{Y}_{101}$, etc.

(k) (5 points) Repeat part (d), using your estimates.

Solution: Using the following code, I found the MSE to be 24.25. This took about 10 minutes to run and it appeared that there was an instability in the estimation at times - the autoregressive component was sometimes estimated to be nonstationary, when included the forecasted data.

```
# Compute 5 one-step MSE for estimated model, using 10000 reps, updating coefs
sqErrors1Alt = rep(0,nSim)
for(j in 1:nSim){
yHat = yDatMat[1:100,j]
for(i in 1:5){
armaEst = arima(yHat,order=c(2,0,5),include.mean=FALSE,method='ML')
gamVecEst = ARMAacf(ar=armaEst$coef[1:2],ma=armaEst$coef[3:7])[2:6]*var(yHat)
g0Est = var(yHat)
g1Est = gamVecEst[1]
g2Est = gamVecEst[2]
g3Est = gamVecEst[3]
g4Est = gamVecEst[4]
gamMatEst = matrix(c(g0Est,g1Est,g2Est,g3Est,g4Est,
     g1Est,g0Est,g1Est,g2Est,g3Est,
  g2Est,g1Est,g0Est,g1Est,g2Est,
  g3Est,g2Est,g1Est,g0Est,g1Est,
  g4Est,g3Est,g2Est,g1Est,g0Est), ncol=5, byrow=TRUE)
beta51Est = solve(gamMatEst)%*%gamVecEst
yHat = c(yHat[1:5]%*%beta51Est,yHat)
}
sqErrors1Alt[j] = (yHat[1] - rev(yDatMat[,j])[1])^2
}
MSE1EstAlt = mean(sqErrors1Alt)
```

(l) (5 points) Repeat parts (e) and (f), using the your estimates.

Solution: Using the following R code, I found the five-step MSE of the estimated $ARMA$ to be 18.30.

```
# Compute exact finite sample 5-step coefs for estimated ARMA, forecast 10000 times
g6Est = 1.3*g5Est - 0.4*g4Est
g7Est = 1.3*g6Est - 0.4*g5Est
g8Est = 1.3*g7Est - 0.4*g6Est
g9Est = 1.3*g8Est - 0.4*g7Est
gamVec5Est = c(g5Est,g6Est,g7Est,g8Est,g9Est)
beta55Est = solve(gamMatEst)%*%gamVec5Est
sqErrors2Est = rep(0,nSim)
for(j in 1:nSim){
yHat105 = rev(yDatMat[96:100,j])%*%beta55Est
sqErrors2Est[j] = (yHat105 - rev(yDatMat[,j])[1])^2
}
MSE2Est = mean(sqErrors2Est)
```

Question 2 . . . . *30 points*

The file ps3Dat.csv contains data on 1 minute returns and order flow for the EUR/USD exchange rate on 13 Nov 2013. The first column of the dataset contains a date/time stamp, the second column reports returns over each minute between 9:30 am and 4:00 pm EST, and the last column reports order flow for the same minutes. Order flow can be thought of as signed volume – trades occurring at the lowest offer prices are counted as a positive number of traded contracts during the time interval, and trades occurring at the highest bid prices are counted as a negative number of traded contracts. Volume, on the other hand, counts all traded contracts positively, regardless of which side of the order book the transactions take place.

(a) (10 points) Estimate a $VAR(2)$ model for EUR/USD returns and order flow. Write the equations of the full model, substituting estimated values for the parameters. (Hint: the estimation can be done equation by equation).

Solution: The general form of the $VAR(2)$ is

$$\begin{bmatrix} R_t \\ O_t \end{bmatrix} = \begin{bmatrix} c_r \\ c_{of} \end{bmatrix} + \begin{bmatrix} \phi^1_{r,1} & \phi^1_{r,2} \\ \phi^1_{of,1} & \phi^1_{of,2} \end{bmatrix} \begin{bmatrix} R_{t-1} \\ O_{t-1} \end{bmatrix} + \begin{bmatrix} \phi^2_{r,1} & \phi^2_{r,2} \\ \phi^2_{of,1} & \phi^2_{of,2} \end{bmatrix} \begin{bmatrix} R_{t-2} \\ O_{t-2} \end{bmatrix} + \begin{bmatrix} \varepsilon_{r,t} \\ \varepsilon_{of,t} \end{bmatrix},$$

where R represents 1-minute EUR/USD returns and O represents 1-minute EUR/USD order flow. The following `R` code estimates each of the individual regressions separately.

```
# Get the data
dat = read.csv('ps4Dat.csv')
rets = dat$Returns
ordFlo = dat$OrderFlow
n = length(rets)

# Estimate VAR(2)
regR = lm(rets[3:n]~rets[2:(n-1)] + rets[1:(n-2)]
           + ordFlo[2:(n-1)] + ordFlo[1:(n-2)])
regOrdFlo = lm(ordFlo[3:n]~rets[2:(n-1)] + rets[1:(n-2)]
                + ordFlo[2:(n-1)] + ordFlo[1:(n-2)])
```

Using the estimates from `R`, we have

$$\begin{bmatrix} R_t \\ O_t \end{bmatrix} = \begin{bmatrix} 8.057e-06 \\ 4.92e+00 \end{bmatrix} + \begin{bmatrix} -5.967e-03 & -1.195e-07 \\ 3.619e+04 & -1.079e-01 \end{bmatrix} \begin{bmatrix} R_{t-1} \\ O_{t-1} \end{bmatrix} + \begin{bmatrix} 6.708e-02 & -4.342e-08 \\ 2.830e+04 & 5.419e-02 \end{bmatrix} \begin{bmatrix} R_{t-2} \\ O_{t-2} \end{bmatrix} + \begin{bmatrix} \varepsilon_{r,t} \\ \varepsilon_{of,t} \end{bmatrix},$$

where

$$\Omega = \mathrm{E}\left[\boldsymbol{\varepsilon}_t \boldsymbol{\varepsilon}_t'\right] = \begin{bmatrix} 5.693e-08 & 2.336e-02 \\ 2.336e-02 & 1.792e+04 \end{bmatrix} \qquad (1)$$

and where the estimates of the covariance matrix values are determined by computing the variances and covariance of the residuals from each regression:

```
# Covariance matrix
omega11 = sum(regRets$resid^2)/383
omega22 = sum(regOrdFlo$resid^2)/383
omega12 = sum(regRets$resid*regOrdFlo$resid)/383
omega21 = omega12
Omega = matrix(c(omega11,omega12,omega21,omega22),ncol=2)
```

(b) (10 points) Rewrite the $VAR(2)$ as a $VAR(1)$, again substituting estimates for parameters.

Solution: The $VAR(2)$ can be recast as a $VAR(1)$ in the following manner:

$$\begin{bmatrix} R_t \\ O_t \\ R_{t-1} \\ O_{t-1} \end{bmatrix} = \begin{bmatrix} 8.057e-06 \\ 4.92e+00 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -5.967e-03 & -1.195e-07 & 6.708e-02 & -4.342e-08 \\ 3.619e+04 & -1.079e-01 & 2.830e+04 & 5.419e-02 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} R_{t-1} \\ O_{t-1} \\ R_{t-2} \\ O_{t-2} \end{bmatrix} + \begin{bmatrix} \varepsilon_{r,t} \\ \varepsilon_{of,t} \\ 0 \\ 0 \end{bmatrix},$$

(c) (10 points) What is the matrix that will orthogonalize the error vector of the $VAR(2)$?

Solution: Recall from the lecture notes that any matrix H which results in $H\Omega H' = D$, where D is some diagonal matrix, will orthogonalize the error term. That is, defining a new error $\boldsymbol{u}_t = H\boldsymbol{\varepsilon}_t$ will result in an error term $\boldsymbol{u}_t$ with diagonal covariance matrix (i.e. no contemporaneous cross correlations). One example of such a matrix would be the transpose of the matrix of eigenvectors of Ω. According to the eigenvalue decomposition,

$$\Omega = T\Lambda T',$$

where Λ is a diagonal matrix of the eigenvalues of Ω and T is the matrix of associated eigenvectors stored as columns. In this case, $T' = T^{-1}$, so that

$$T'\Omega T = \Lambda.$$

Thus, T' is a matrix which will orthogonalize the error term. We find this matrix in `R` in the following manner:

```
# Orthoganlize
eigOut = eigen(Omega)
Lambda = diag(eigOut$values)
Tmat = eigOut$vectors
```

The resulting matrix is

$$T' = \begin{bmatrix} 1.303e-06 & 1 \\ -1 & 1.303e-06 \end{bmatrix}.$$