Lancaster University
Management School

Group No: 11

Department of Accounting and Finance

Group Coursework Cover Sheet

Coursework submitted for assessment will only be marked if the following signed declaration sheet accompanies it.

Assessed Coursework Declaration

I confirm that I have read and understood the University's regulations relating to Plagiarism (as stated in the courses handbook). I declare that this submission is my own work. I have not submitted it elsewhere in substantially the same form towards the award of a degree or other qualification. It has not been written or composed by any other person and all sources have been appropriately referenced and acknowledged. I agree that this work may be interrogated for plagiarism software.

SURNAME	FORENAME	Library Card Number(s)	Signature(s)
de Vibe	Emily	35275809	Emily de Vibe

Module Number: ACF629 **Tutor:**

Submission Date: 20.03.20.

Students may be asked to submit themselves to an oral examination on the basis of any item of coursework. Students should note that a mark of zero may be given to any coursework that is found to be in breach of university regulations. Any cases of plagiarism may be brought to the attention of the Board of Examiners. Breaches of regulations may be referred to in any references which members of the Department are asked to provide.

The following penalties will be applied to all coursework that is submitted after the specified submission date:

Up to 3 days late	deduction of 10 marks
Beyond 3 days late	no marks awarded

FOR OFFICE USE ONLY

Extension: Yes / No Date Received:

Penalty (if applicable): **Extension Date:**

Final Mark % (after deduction of penalty if applicable):

Emily de Vibe

Student nr. 35275809

ACF629 -Financial Econometrics

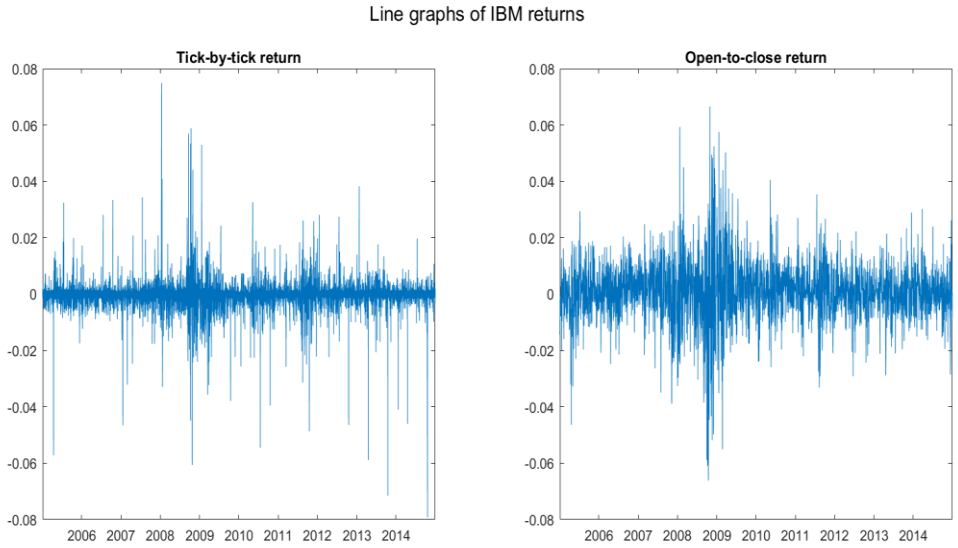
Coursework Assignment 2020

Total nr of pages: 10

Part 1: ARMA-GARCH Model

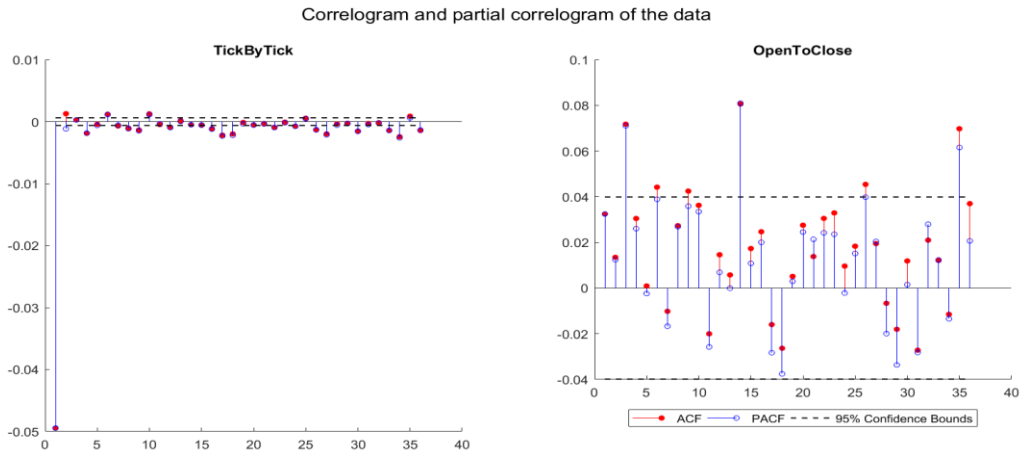
a) I chose to work with the IBM stock. The matlab code for computing the tick-by-tick and the open-to-close (log) return series for each trading day in the dataset is presented in the appendix p. 11.

b) **Fig 1.1**



From the line graphs of IBM returns in Fig. 1.1 above it seems like both tick-by-tick returns and open-to-close returns are stationary with mean approximately equal to 0. Neither returns are very persistent, which is confirmed by plotting the autocorrelation function (ACF) for both returns as illustrated in Fig. 1.2 below, which shows that most autocorrelation coefficients are not significant as the nr of lags increases.

Fig. 1.2



Neither returns series look normally distributed which can be observed from drawing a histogram of both returns as illustrated in Fig. 1.3 on the next page. In addition, by performing a Jarque-Bera test, we obtain in both cases a p-value of 0 , which means we reject the null-hypothesis in favour of the alternative one. Hence both returns are not normally distributed. Furthermore, both returns are left-skewed and more heavy-tailed compared to the normal distribution, but the tick-by-tick returns

possesses these characteristics to a much greater extent than the open-to-close returns (skewness of -7.45 vs -0.25 and kurtosis of 6533.7 vs 7.7745).

Fig 1.3

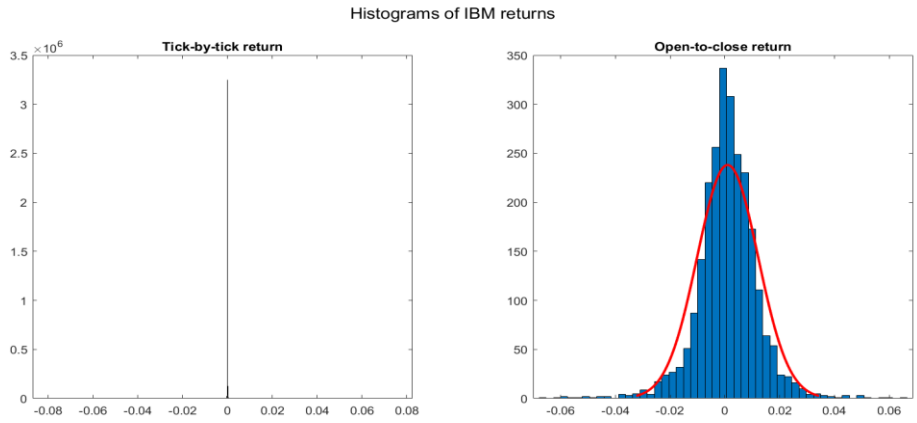


Fig 1.4 below shows the autocorrelation function (ACF) for the tick-by-tick and the open-to-close squared returns. We can see that the coefficients for the ACF for both returns are significant also at long lags, providing strong predictability of volatility. However, for the tick-by-tick returns, the value of the coefficients for the ACF are closer to 0 and decline more rapidly towards an insignificant level as the nr of lags increases, while for the open-to-close returns, the value of the coefficients of the ACF are greater and remain consistently significant as the nr of lags increases. This indicates that the volatility of the open-to-close returns have a greater level of persistence for a longer period of time compared the volatility of the tick-by-tick returns. This is again illustrated in the line graph of the open-to-close returns in Fig. 1.1, which cluster more intensively together than for the tick-by-tick returns.

Fig. 1.4

As a conclusion, we observe that both tick-by-tick and open-to-close returns possess the same characteristics in terms of the stylized facts of financial data, such as for instance volatility clustering and a more heavy-tailed and left-skewed distribution compared to the normal distribution. We also

see that when the observation frequency increases from open-to-close to tick-by-tick returns, these financial characteristics do become more or less intensified. For example, the level of volatility clustering is more present with the open-to-close returns, while the tick-by-tick returns have a greater kurtosis and negative skewness compared to the open-to-close returns.

c) First of all, I need to check whether the daily open-to-close (log) return time series computed in (a) is stationary to find the best model in the general ARMA-GARCH modelling framework. The Augmented Dickey-Fuller (ADF) test with automatic lag selection based on an information criterion from BIC is therefore applied (without differencing the returns. Three sets of tests are performed for this test: the ADF with no drift or trend, with a drift and with both drift and trend. As shown in Table 1.1 below, these tests are all rejected (p-value: 0.001) meaning that the open-to-close returns are a covariance-stationary AR(1) process. However, by observing the line-graph of the open-to-close returns (Fig 1.1) , we observe that the open-to-close returns do not grow with a linear nor polynomial growth rate, which implies it is unlikely that our best-fit model should include a drift or a trend.

Table 1.1 ADF test with optimal lag selection using BIC

	None	*Drift*	*Trend*
Test Stat	-34.09	-34.463	-34.467
10 % C.V	-1.6168	-2.5683	-3.1283
5 % C.V.	-1.9416	-2.8642	-3.4141
1 % C.V.	-2.5689	-3.4366	-3.9675
P-value	0.001	0.001	0.001
Lag used	0	0	0

The Phillips-Perron Test is also performed to check that we obtain the same results as for the ADF test. Again this involves completing three tests for the same different model specifications as the ADF test (no drift or trend, with a drift and with both drift and trend), but here we use a fixed bandwidth for the Newey-West type covariance estimates (equal to 22). Since we obtain the same results as for the ADF test (rejection of the null hypothesis in favour for the alternative one for each three tests), we can conclude that the open-to-close returns must be covariance stationary.

Table 1.2 PP test with Newey-West Bandwidth

	None	*Drift*	*Trend*
Test Stat	-49.893	-49.608	-49.1283
10 % C.V	-1.6168	-2.5683	-3.1283
5 % C.V.	-1.9416	-2.8642	-3.4141
1 % C.V.	-2.5689	-3.4366	-3.9675
P-value	0.001	0.001	0.001

Since the optimal lag chosen from using the ADF test was 0, we then move on to pick the best model by looping over all models starting from an ARMA(0,0) model to an ARMA(1,1) model (a total of 4 models) based on information criteria (BIC). By computing the log-likelihood and information criteria for all of these models, the lag order p and q with the lowest information criterion is p = 1 and q = 1 or an ARMA(1,1) model. By estimating an ARMA(1,1) for the open-to-close returns and constructing t-statistics and corresponding values we obtain table 1.3 below. From this table we can see that for the estimated ARMA(1,1) model, both $\phi_1 = 0.92247$ and $\theta_1 = -0.8914$ are statistically significant, but the constant c is not significant at the 0.01 level. Hence this parameter could be set to 0.

Table 1.3 ARIMA(1,0,1) model (Gaussian distribution)

	Value	*Standard Error*	*T- Statistic*	*P-value*
C	8.0012e-05	3.267e-05	2.4491	0.014322
ϕ_1	0.92247	0.024813	37.177	0
θ_1	-0.8914	0.030111	-29.603	0
σ^2	0.00012286	1.9245e-06	63.841	0

If we then compute the residuals of our ARMA(1,1) model and performing an Engle test for residual heteroscedasticity, we obtain a p-value of 0. Hence we reject the null hypothesis of no conditional heteroscedasticity and conclude that there are significant ARCH effects in the open-to-close returns that our ARMA(1,1) model does not capture. In other words, we need to implement an ARMA-GARCH model in order to capture these ARCH effects.

We therefore continue by computing the optimal GARCH lags based again on the information criterion (BIC). By looping over each model from an ARMA(1,1)-GARCH(0,0) to an ARMA(1,1)-GARCH(2,2) model and calculating the corresponding information criterion matrix, the optimal GARCH and ARCH lags (with the lowest information criterion) is p = 1 and q = 1 respectively. We therefore need to estimate an ARMA(1,1)-GARCH(1,1) model, for which the estimated parameters and test-statistics are show in table 1.4 and 1.5 below.

Table 1.4 ARIMA(1,0,1) model (Gaussian distribution)

	Value	*Standard Error*	*T-statistic*	*P-value*
C	0.00014307	5.6657e-05	2.5252	0.011563
ϕ_1	0.88452	0.04154	21.293	0
θ_1	-0.84785	0.04756	-17.827	0

Table 1.5 GARCH(1,1) conditional variance model (Gaussian distribution)

	Value	*Standard Error*	*T-statistic*	*P-value*
ω	3.0732e-06	7.9474e-07	3.867	0.00011019
β_1	0.87601	0.012328	71.058	0
α_1	0.095708	0.009778	9.7881	0

We can see from the two tables above that the all the estimated parameters for the ARMA(1,1)-GARCH(1,1) model are significant except for the constant c in the ARMA(1,1) model which is not statistically significant at the 0.01 level. Furthermore, we have that

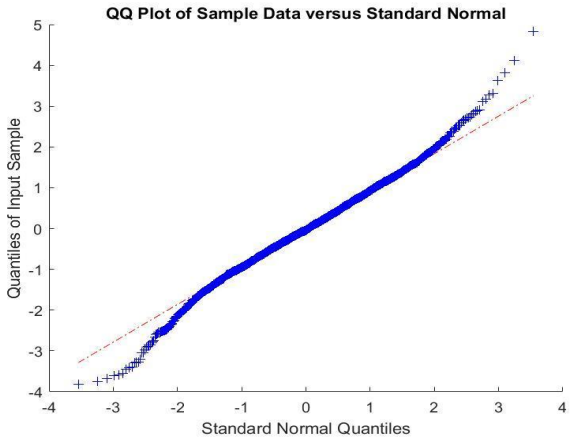
$$\beta_1 + \alpha_1 = 0.87601 + 0.095708 = 0.971718$$

which is very close to 1, suggesting that the volatility process is very persistent.

As a conclusion, the ARMA-GARCH(1,1) model is the best model for the daily open-to-close return time series, but we need to perform diagnostic checks to see whether we can further improve our selected best fitted model.

d) We start by inferring the residuals from our fitted ARMA(1,1)-GARCH(1,1) model and then compute the standardized residuals. We compare the quantiles of our standardized residuals against the quantiles of the normal distribution as shown in Fig. 1.6 below. We can see that our residuals are not normally distributed, which is confirmed by performing a Jarque-Bera test (p-value: 1.0000e-03).

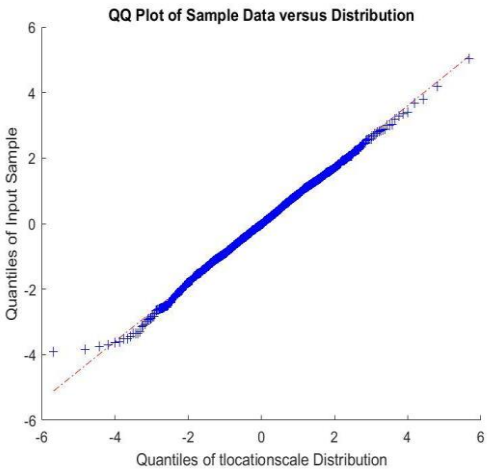
Fig. 1.6



But if we perform an Engle test for residual heteroscedasticity, we obtain a p-value of 0, so that our fitted model captures the ARCH effects present in the open-to-close log returns.

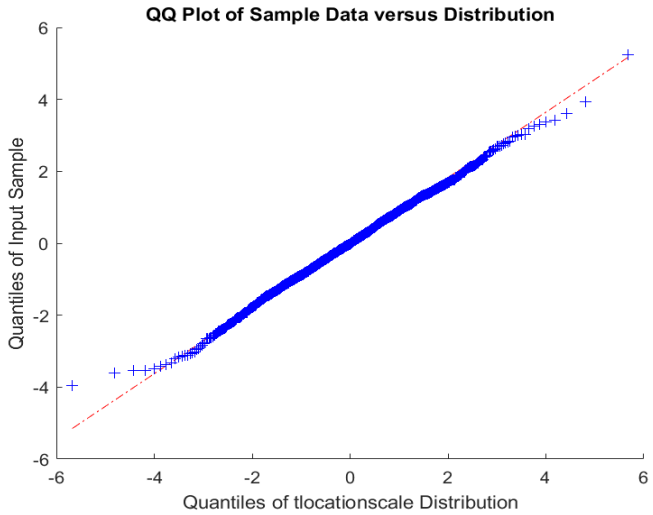
If we instead try to estimate the corresponding t-distributed GARCH model, we can compute the fitted standardized residuals which we can compare with the quantiles of the t-distribution using a QQ-plot, as shown in Fig. 1.7 below.

Fig. 1.7



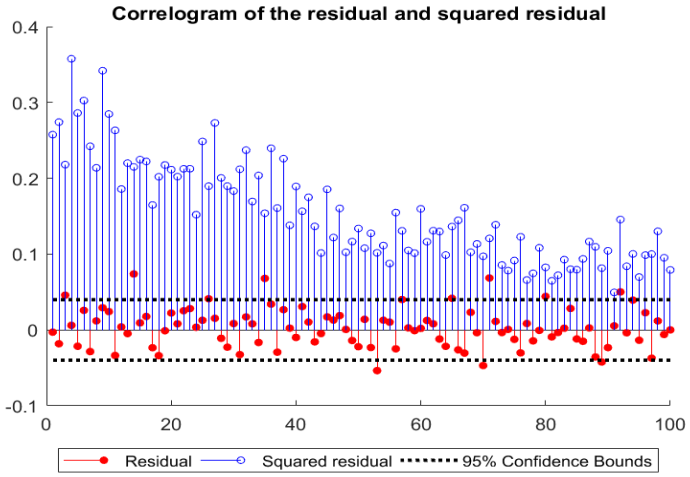
We can see that the goodness-of-fit is much better for the t-distribution than the normal distribution. We further consider asymmetric GARCH models such as the GJR-GARCH(1,1) model specified with a t-distribution. By computing the standardized residuals and comparing these with t-distribution using a QQ-plot as shown in the figure below, we obtain a very similar QQ-plot to the one we obtain with the t-distributed ARMA(1,1)-GARCH(1,1) model.

Fig. 1.8



Furthermore, since the GJR-GARCH model nests the GARCH model, we can conduct a likelihood-ratio test to see if the extra parameter in the GJR-GARCH model improves the likelihood significantly. The p-value obtained is 0 and is therefore significant at any conventional level, which suggests we should use the asymmetric structure of GJR-GARCH model to fit the open-to-close returns.

Fig 1.9



Finally, fig. 1.9 shows the autocorrelation function for the residuals and the squared residuals, showing that the autocorrelation of the residuals is not significant.

Therefore the ARMA(1,1)-GJR(1,1) model is the best model to fit the open-to-close returns, for which the estimated parameters along with the corresponding t-statistics are provided in the two tables below. We can see from these table that the estimated constant c from the ARMA(1,1) model and the estimated ARCH{1} coefficient α_1 from the GJR(1,1) model are not significant at the 0.01 level.

Table 1.6 ARIMA(1,0,1) model (t Distribution)

	Value	*Standard Error*	*T-statistic*	*P-value*
C	0.00013195	5.956e-05	2.2155	0.026729
$\boldsymbol{\phi_1}$	0.87059	0.051299	16.971	0
$\boldsymbol{\theta_1}$	-0.84175	0.057336	-14.681	0
DoF	9.1731	1.6395	5.595	2.2063e-08

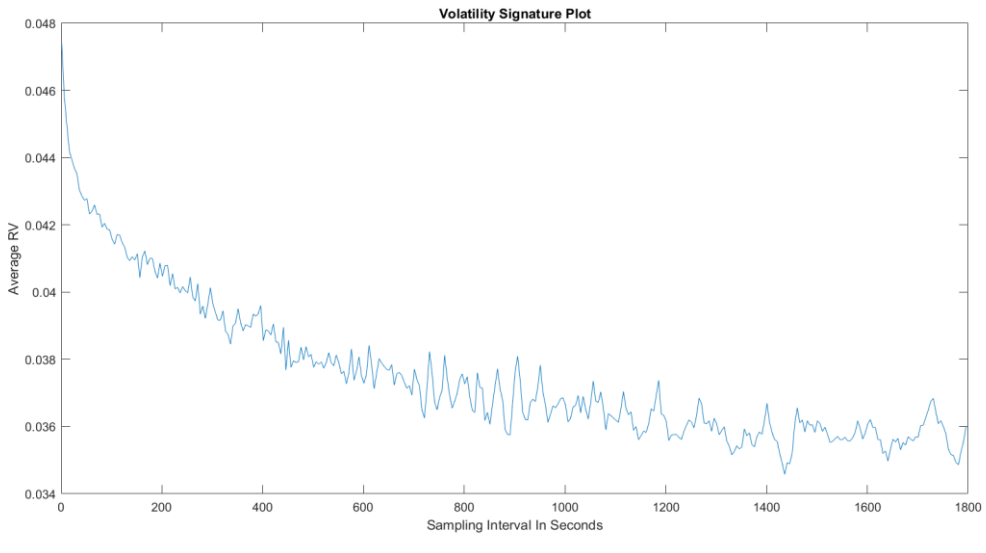
Table 1.7 GJR(1,1) conditional variance model (t distribution)

	Value	*Standard Error*	*T-statistic*	*P-value*
$\boldsymbol{\omega}$	2.5798e-06	8.5835e-07	3.0055	0.0026511
$\boldsymbol{\beta_1}$	0.89248	0.014831	60.178	0
$\boldsymbol{\alpha_1}$	0.032586	0.013487	2.416	0.015691
Leverage	0.099503	0.020197	4.9267	8.3639e-07
DoF	9.1731	1.6395	5.595	2.2063e-08

Part 2: Realized Volatility and Forecasting

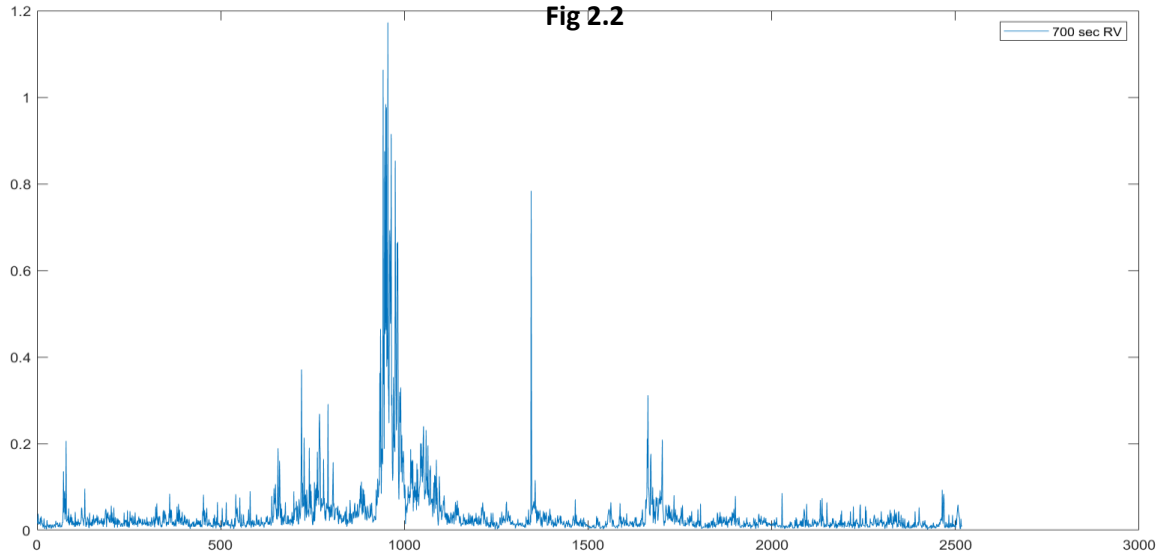
e) By constructing a volatility signature plot with daily annualized realized volatility estimators based on the high frequency data for the chosen stock, we obtain the figure below.

Fig. 2.1



f) We select an optimal sampling frequency of 700 seconds as this is approximatively the sampling interval at which the average RV starts to stabilize itself. Then we construct the daily annualized volatility estimator based on this optimal sampling frequency as show in Fig 2.2 below and as presented in the appendix p. 24.

Fig 2.2



g) The matlab code for this task is presented in the appendix p. 25.

h) To evaluate our forecasting performances we can compute the MSE, MAPE and QLIKE measures of the forecasts as shown in the table below.

Table 2.1 Evaluation of forecasting performances

	MAPE	*MSPE*	*QLIKE*
RV	0.0086515	0.00015263	0.30441
GJR-GARCH-t	0.011787	0.00027487	0.5434

All three loss functions suggest that the GJR-GARCH-t measure performs the worst, which is expected as the GARCH model does not take intraday price information into account.

Then we compute the forecasting errors so that we can use a modified Diebold & Mariano (DM) test to test if one model significantly outperforms the other, where we will be comparing the HAR-RV against the GJR-GARCH-t model. Our results are illustrated in the table below.

Table 2.2 DM test

	Test statistic	*p-value*
GJR-GARCH-t vs RV	-6.0161	3.4415e-09

We can see from Table 2.2 that the obtained test statistic is negative, which means the HAR-RV model outperforms the HAR-GARCH model. Because the null-hypothesis of the DM test is rejected, the HAR-GARCH model performs significantly worse than the HAR-RV model.

APPENDIX

Below follows the matlab code used to solve this coursework.

```
% Coursework

% Part 1: ARMA-GARCH Model
% a) For your chosen stock, compute the tick-by-by tick log-return
and the open-to-close log return series for each trading day in the
dataset.

clearvars; clc;

% set working directory
% cd('C:\Users\Eier\Documents\Lancaster\ACF609\Project')

%load csv file into MATLAB as table
dmat=readtable('IBM_trade_quote.csv');

% extract data
data=dmat{:,1:4};

size(dmat); % 10535140           6

%extract all dates
dateTbT = data(:,2); % 10535140 x 1 vector

% convert dates to str
dateTbT_str = num2str(dateTbT);

% convert str to datetime
dateTbT_dt =
datetime(dateTbT_str,'InputFormat','yyyyMMdd','Format','dd.MM.yyyy')
;

%Extract unique day counters
dc=unique(data(:,1)); % 2517 x 1

% Extract unique dates
ddate=unique(data(:,2));
% ddate: 2517 x 1 matrix

% convert unique dates to string
dateOtC_str = num2str(ddate);

% convert unique dates from string to datetime format
dateOtC_dt =
datetime(dateOtC_str,'InputFormat','yyyyMMdd','Format','dd.MM.yyyy')
;


% initializing the open-to-close log return vector
retOtC=zeros(length(dc),1);
```

```
% initializing the tick-by-tick log return vector
retTbT=diff(log(data(:,4)));

%For loop for all trading days
for i=1:length(dc)
   pricet=data(data(:,1)==dc(i),3:4);

   % collecting open-to-close log returns
   retOtC(i)=log(pricet(end,2))-log(pricet(1,2));

end

% length(retTbT) % 10535139 x 1
% length(retOtC) % 2517 x 1

% b) Provide a descriptive analysis for both return series computed
above.
% Discuss your findings and relate them to the stylized facts of
financial
% data.

%get the size of the data
[rowsTbT, colsTbT] = size(retTbT); % [ 10535139, 1 ]
[rowsOtC, colsOtC]=size(retOtC); % [2517,1]

names= {'TickByTick','OpenToClose'};

%Computing the statistics

%mean
meandata= [mean(retTbT), mean(retOtC)];

%variance
vardata= [var(retTbT), var(retOtC)];

% minimum
mindata= [min(retTbT), min(retOtC)];

% maximum
maxdata = [max(retTbT),max(retOtC) ];

%standard deviation.
stddata= [std(retTbT), std(retOtC) ];

% skewness = the sample skewness of X
skedata= [ skewness(retTbT), skewness(retOtC)];

%kurtosis
% sample kurtosis
kurdata=[kurtosis(retTbT), kurtosis(retOtC) ];

%quantiles
qvector=[.01 .05 .1 .5 .9 .95 .99]; % vektor
qdata=[quantile(retTbT,qvector); quantile(retOtC,qvector)];
```

```
%Prepare a table for the results
%combine the statistics first:
stat=[meandata; vardata; maxdata;mindata; stddata; skedata;
kurdata;qdata'];

%convert it to table with proper titles
stattable=array2table(stat,'VariableNames',names,'RowNames', {'Mean'
...
    'Variance' 'Maximum' 'Minimum' 'Std.Dev.' 'Skewness' 'Kurtosis'
...
    'Q(0.01)' 'Q(0.05)' 'Q(0.1)' 'Q(0.5)' 'Q(0.9)' 'Q(0.95)'
'Q(0.99)'});

%display the table to the output window
disp(stattable);
%                TickByTick     OpenToClose
%                ___________    ___________
%
%    Mean         4.5862e-08    0.00097579
%    Variance     6.0638e-08    0.00012392
%    Maximum        0.074876      0.066563
%    Minimum       -0.079237     -0.066163
%    Std.Dev.     0.00024625      0.011132
%    Skewness        -7.4979      -0.24509
%    Kurtosis         6533.7        7.7745
%    Q(0.01)     -0.00060646     -0.031155
%    Q(0.05)     -0.00029193     -0.016138
%    Q(0.1)      -0.00019445     -0.010567
%    Q(0.5)                0    0.00086823
%    Q(0.9)       0.00019389      0.012998
%    Q(0.95)      0.00029468      0.017482
%    Q(0.99)      0.00061121      0.029684


% line graphs of returns
plotsize=[100,100,1200,550];
fig1=figure;
subplot(1,2,1);
plot(dateTbT_dt(2:end),diff(log(data(:,4))));
title('Tick-by-tick return');
subplot(1,2,2);
plot(dateOtC_dt,retOtC);
title('Open-to-close return');
suptitle('Line graphs of IBM returns');

% histograms
fig1.Position=plotsize;
fig2=figure;
subplot(1,2,1);
histogram(retTbT);
title('Tick-by-tick return');
subplot(1,2,2);
histfit(retOtC);
title('Open-to-close return');
suptitle('Histograms of IBM returns');
fig2.Position=plotsize;
```

```
%Correlogram and partial correlogram up to lag L
L=36;
fig3=figure;
Q=zeros(L,2);
[acfTbT,lag1TbT,bound1TbT]=autocorr(retTbT,L);
[pacfTbT,lag2TbT,bound2TbT]=parcorr(retTbT,L);
%compute Ljung-Box test statistics
Q(:,1)=length(retTbT)*(length(retTbT)+2)*cumsum((acfTbT(2:L+1).^2)./
(length(retTbT)-lag1TbT(2:L+1)));
subplot(1,2,1);
hold on;
p1=stem(1:1:L,acfTbT(2:L+1),'filled'); %ploting acf
p2=stem(1:1:L,pacfTbT(2:L+1)); %ploting pacf
p=plot(1:1:L,[ones(L,1).*bound1TbT(1) ones(L,1).*bound1TbT(2)]);
%ploting confidence bounds
hold off;
%Graph settings
p1.Color='r';
p1.MarkerSize=4;
p2.Color='b';
p2.MarkerSize=4;
p(1).LineWidth=1;
p(1).LineStyle='--';
p(1).Color='k';
p(2).LineWidth=1;
p(2).LineStyle='--';
p(2).Color='k';
title('Tick-by-tick return');

[acfOtC,lag1OtC,bound1OtC]=autocorr(retOtC,L);
[pacfOtC,lag2OtC,bound2OtC]=parcorr(retOtC,L);
%compute Ljung-Box test statistics
Q(:,2)=length(retOtC)*(length(retOtC)+2)*cumsum((acfOtC(2:L+1).^2)./
(length(retOtC)-lag1OtC(2:L+1)));
subplot(1,2,2);
hold on;
p1=stem(1:1:L,acfOtC(2:L+1),'filled'); %ploting acf
p2=stem(1:1:L,pacfOtC(2:L+1)); %ploting pacf
p=plot(1:1:L,[ones(L,1).*bound1OtC(1) ones(L,1).*bound1OtC(2)]); %
ploting confidence bounds
hold off;
%Graph settings
p1.Color='r';
p1.MarkerSize=4;
p2.Color='b';
p2.MarkerSize=4;
p(1).LineWidth=1;
p(1).LineStyle='--';
p(1).Color='k';
p(2).LineWidth=1;
p(2).LineStyle='--';
p(2).Color='k';
title('Open-to-close return');
%legend settings
```

```
legend('ACF', 'PACF','95% Confidence Bounds',
'Location','southoutside','Orientation','horizontal');
suptitle('Correlogram and partial correlogram of IBM returns');
fig3.Position=plotsize;

% ACF for IBM squared returns
L=36;
fig4=figure;
[acfTbT,lag1TbT,bound1TbT]=autocorr(retTbT.^2,L);
subplot(1,2,1);
hold on;
p1=stem(1:1:L,acfTbT(2:L+1),'filled'); %ploting acf
p=plot(1:1:L,ones(L,1).*bound1TbT(1)); %ploting confidence bound
hold off;
%Graph settings
p1.Color='r';
p1.MarkerSize=4;
p.LineWidth=1;
p.LineStyle='--';
p.Color='k';
title('tick-by-tick squared returns');

[acfOtC,lag1OtC,bound1OtC]=autocorr(retOtC.^2,L);
subplot(1,2,2);
hold on;
p1=stem(1:1:L,acfOtC(2:L+1),'filled');
p=plot(1:1:L,ones(L,1).*bound1OtC(1));
hold off;
%Graph settings
p1.Color='r';
p1.MarkerSize=4;
p.LineWidth=1;
p.LineStyle='--';
p.Color='k';
title('Open-to-close squared returns');
%legend settings
legend('ACF','95% Upper Confidence Bound',
'Location','southoutside','Orientation','horizontal');
suptitle('ACF of IBM squared returns');
fig4.Position=plotsize;

% ------------------------------------------------------------
%Jacque-Bera test for normality
% H0: Normal
JB = [(length(retTbT))/6*(skedata(1).^2+0.25*(kurdata(1)-3).^2),
(length(retOtC))/6*(skedata(2).^2+0.25*(kurdata(2)-3).^2)]; % data:
log returns
JBpvalue=chi2cdf(JB,2,'upper'); % p-verdi

%Ljung-Box test for correlation (pvalue computed using the Q
obtained from the
%correlogram)
LBpvalue=chi2cdf(Q,repmat((1:1:L)',1,2),'upper');
teststat=[ JB;Q];
pstat=[ JBpvalue; LBpvalue];
testtable=cell(L+1,3);
```

```
%Use * to indicate significance: ***: significant at 1%, **:
significant at
%5%, *: significant at 10%


for i=1:L+1
    if i==1
        testtable(i,1)={'JB stat'};
    else
        testtable(i,1)={strcat('LB(', num2str(i-1,'%5.0f'),')')};

    end
    for j=2:3
        %  p value <= alpha: reject H0
        if pstat(i,j-1)<=0.01  % H0 blir forkastet hvis alpha = 0.01
            testtable(i,j)={strcat(num2str(teststat(i,j-
1),'%5.2f'),'***')};% test verdi og ikke p value

% settes inn
        elseif pstat(i,j-1)<=0.05
              testtable(i,j)={strcat(num2str(teststat(i,j-
1),'%5.2f'),'**')};
        elseif pstat(i,j-1)<=0.1
              testtable(i,j)={strcat(num2str(teststat(i,j-
1),'%5.2f'),'*')};
        else
              testtable(i,j)={num2str(teststat(i,j-1),'%5.2f')}; %
H0 blir ikke forkastet
                                                                  %
ingen
                                                                  %
stjerner
                                                                  %
        end
    end
end
%create table of results
testtable=cell2table(testtable(:,2:end), 'VariableNames',  names,
'RowNames',testtable(:,1));

disp(testtable);

%                      TickByTick          OpenToClose
%                  ____________________    ___________
%
%    JB stat       '18722109883869.32***'    '2415.96***'
%    LB(1)         '25761.20***'             '2.65'
%    LB(2)         '25778.43***'             '3.11'
%    LB(3)         '25779.20***'             '16.09***'
%    LB(4)         '25816.37***'             '18.43***'
%    LB(5)         '25818.34***'             '18.43***'
%    LB(6)         '25833.15***'             '23.36***'
%    LB(7)         '25838.26***'             '23.62***'
%    LB(8)         '25850.35***'             '25.50***'
%    LB(9)         '25869.65***'             '30.06***'
%    LB(10)        '25886.20***'             '33.37***'
```

```
%    LB(11)      '25888.52***'              '34.39***'
%    LB(12)      '25897.17***'              '34.93***'
%    LB(13)      '25897.37***'              '35.01***'
%    LB(14)      '25900.00***'              '51.46***'
%    LB(15)      '25902.92***'              '52.22***'
%    LB(16)      '25916.81***'              '53.76***'
%    LB(17)      '25967.60***'              '54.41***'
%    LB(18)      '26009.23***'              '56.17***'
%    LB(19)      '26009.44***'              '56.23***'
%    LB(20)      '26012.37***'              '58.16***'
%    LB(21)      '26013.39***'              '58.64***'
%    LB(22)      '26022.32***'              '61.00***'
%    LB(23)      '26022.41***'              '63.76***'
%    LB(24)      '26028.16***'              '63.99***'
%    LB(25)      '26031.30***'              '64.85***'
%    LB(26)      '26050.02***'              '70.09***'
%    LB(27)      '26091.34***'              '71.05***'
%    LB(28)      '26092.77***'              '71.17***'
%    LB(29)      '26093.54***'              '72.00***'
%    LB(30)      '26118.14***'              '72.36***'
%    LB(31)      '26119.09***'              '74.24***'
%    LB(32)      '26119.44***'              '75.36***'
%    LB(33)      '26140.17***'              '75.74***'
%    LB(34)      '26203.04***'              '76.08***'
%    LB(35)      '26211.03***'              '88.51***'
%    LB(36)      '26232.15***'              '92.00***'


% c) For the daily open-to-close return time series computed in a), find
% the best model in the general ARMA-GARCH modelling framework. You should
% provide  detailed model selection procedures and justify your choice
% Use a sensible set of competing ARMA-GARCH specifications.

%Set the significance level of the test
alpha=[0.1,0.05,0.01];

%Augmented Dickey-Fuller test with automatic lag selection based on
%information criterion.

sel_method='BIC';

dfout=ADF_auto(retOtC,0,30,sel_method);% dif = 0

%For PP and KPSS test, the selection of lags is more involved. Here I used
%a fixed bandwith for the Newey-West type variance-covariance estimates,
%which may not be the best selection method for the two type of tests.
Bandwidth=floor(4* (length(retOtC)^(2/9)) ); % 22

%Philip Perron test
```

```
%Perform three sets of tests for three different model
specifications
[h1,pValue1,stat1,cValue1,reg1]=pptest(retOtC,'Model','AR', 'alpha',
alpha,'lags', Bandwidth);
[h2,pValue2,stat2,cValue2,reg2]=pptest(retOtC,'Model','ARD',
'alpha',alpha, 'lags', Bandwidth);
[h3,pValue3,stat3,cValue3,reg3]=pptest(retOtC,'Model','TS',
'alpha', alpha,'lags', Bandwidth);

%Prepare a table to present the test outputs
pptable=[stat1(1) cValue1 pValue1(1); stat2(1) cValue2 pValue2(1)
;stat3(1) cValue3 pValue3(1)];
ppout=array2table(pptable','RowNames', {'Test Stat' ' 10% C.V.' '5%
C.V.' '1% C.V.' 'P-value' },...
'VariableNames', {'None' 'Drift' 'Trend'});

%Print the results in the command window
disp(strcat('ADF test with optimal lag selection using', {' '},
sel_method));
% ADF test with optimal lag selection using BIC'
disp(dfout);

 %                 None        Drift       Trend
 %                ________    ________    ________
%
%    Test Stat     -34.09     -34.463     -34.467
%    10% C.V.     -1.6168     -2.5683     -3.1283
%    5% C.V.      -1.9416     -2.8642     -3.4141
%    1% C.V.      -2.5689     -3.4366     -3.9675
%    P-value        0.001       0.001       0.001
%    Lag used           0           0           0

disp('PP test with Newey-West fixed Bandwidth');
% PP test with Newey-West fixed Bandwidth
disp(ppout);
%                   None        Drift       Trend
%                  ________    ________    ________
%
%    Test Stat     -49.839     -49.608     -49.593
%    10% C.V.     -1.6168     -2.5683     -3.1283
%    5% C.V.      -1.9416     -2.8642     -3.4141
%    1% C.V.      -2.5689     -3.4366     -3.9675
%    P-value        0.001       0.001       0.001

T=length(retOtC); % 2517

%Picking the best model from ARMA(0,0) to ARMA(pmax,qmax) based on
information
%criteria

%initiallizing information criteria mat
pmax=1;
qmax=1;
AICmat=zeros(pmax+1,qmax+1);
BICmat=zeros(pmax+1,qmax+1);
HQICmat=zeros(pmax+1,qmax+1);
```

```
%Compute the log-likelihood and information criteria for the models
considered
%Note that I started with ARMA(0,0) and loop over all the models to
%ARMA(pmax,qmax).
for i=1:pmax+1 % pmax = 0,1
    for j=1:qmax+1 % qmax = 0,1
        Mdltemp=arima(i-1,0,j-1);
        [~,~,logL,~] = estimate(Mdltemp,retOtC,'Display','off');

        %You can also use the aicbic function to compute AIC or BIC.
        AICmat(i,j)=-2*logL/T+2*(-1+i+j)/T;
        % i+j-1
        BICmat(i,j)=-2*logL/T+(-1+i+j)*log(T)/T;
        HQICmat(i,j)=-2*logL/T+2*(-1+i+j)*log(log(T))/T;
    end
end

%Find the location of the minimum information critieria.
[paic,qaic]=find(AICmat==min(min(AICmat)));

[pbic,qbic]=find(BICmat==min(min(BICmat)));
[phqic,qhqic]=find(HQICmat==min(min(HQICmat)));

%Choose best model using BIC for example
pbest=pbic-1; % 1
qbest=qbic-1; % 1

% [P,Q] = ARMA_optimal(retOtC,pmax,qmax,'BIC');% P = 1, Q = 1

Mdlbest=arima(pbest,0,qbest);
[EstMdlbest,EstParamCov,logL1,info] = estimate(Mdlbest,retOtC);
%EstMdlbest: model containing parameter estimates

% ARIMA(1,0,1) Model (Gaussian Distribution):

%              Value       StandardError    TStatistic      PValue
%            _________    _____________    __________
_____________
%  Constant    8.0012e-05      3.267e-05        2.4491
0.014322
%  AR{1}          0.92247       0.024813        37.177
0
%  MA{1}          -0.8914       0.030111       -29.603
0
% Variance    0.00012286     1.9245e-06        63.841
0

%The parameter estimates are stored in the structure EstMdlbest. To
retrieve
%them, use the following command:
bhat=[EstMdlbest.Constant; EstMdlbest.AR{1};  EstMdlbest.MA{1};
EstMdlbest.Variance   ];

% The standard errors can be retrieved from the EstParamCov matrix
se=sqrt(diag(EstParamCov));
```

```
%construct t-statistics and p-values ourselves:
tstat=bhat./se; % tstat = estimated value of parameters/
SE(parameters)

pvalue=(1-normcdf(abs(tstat)))*2;
%Prepare a table for the estimation results ourselves.
armadata=[bhat se tstat pvalue ];
armaout=array2table(armadata, 'RowNames', {'c' 'phi' 'theta'
'sigmasq'}, 'VariableNames', {'bhat' 'SE' 'tstat' 'pvalue'});
disp(armaout);
%                bhat           SE          tstat       pvalue
%              __________    __________    _______    _________
%
%    c          8.0012e-05     3.267e-05     2.4491    0.014322
%    phi           0.92247      0.024813     37.177           0
%    theta         -0.8914      0.030111    -29.603           0
%    sigmasq    0.00012286    1.9245e-06     63.841           0

% Residual analysis
[E,V,logL2]=infer(EstMdlbest,retOtC);

%ARCH LM test to check for potential heteroscedasticity in the
error
%term. We only check for the first lag.
 [h,pValue,stat,cValue] = archtest(E,'Lags',1);
  % h = 1
  % pValue = 0


%Compute optimal GARCH lags using a function GARCH_optimal.m.
[gbest,abest,ICmat] = GARCH_optimal(retOtC, 2, 2,1, 1, 'BIC');
% gbest = 1
% abest = 1

%Using the optimal GARCH lags
Mdl=arima('ARLags',1','MALags',1,'Variance',garch(gbest,abest));

%Estimate the model
[estmdl, estParamCov,logL, info] =estimate(Mdl,retOtC);

% ARIMA(1,0,1) Model (Gaussian Distribution):

%                 Value       StandardError    TStatistic
PValue
%              __________    _____________    __________
___________
%
%    Constant    0.00014307     5.6657e-05        2.5252
0.011563
%    AR{1}          0.88452        0.04154        21.293
0
%    MA{1}         -0.84785        0.04756       -17.827
0
```

```
%    GARCH(1,1) Conditional Variance Model (Gaussian Distribution):

%                  Value       StandardError    TStatistic
PValue
%                 _________    _____________    __________
_________

%    Constant    3.0732e-06     7.9474e-07         3.867
0.00011019
%    GARCH{1}       0.87601       0.012328         71.058
0
%    ARCH{1}       0.095708       0.009778         9.7881
0
```

% d) Perform diagnostic checks on your best ARMA-GARCH model. Discuss the
% goodness-of-fit of your model.

```
% residuals for the ARMA(1,1)-GARCH(1,1) fitted model
[E,V,logl]=infer(estmdl,retOtC);

%Compute fitted standardized error term
z=E./sqrt(V);

%Diagnostic testing
%testing against a standard normal distribution
qqplot(z);
% Jarque Bera test
[h,p,stat,cv]=jbtest(z); % testing standardized residuals
% h: 1 => not normally distributed
% p: 1.0000e-03

%Detecting remaining ARCH effects
[h2,p2,stat2,cv2] = archtest(z,'Lags',1:10);
% h2: 0   0   0   0   0   0   0   0   0   0 0
% p:   0     0     0     0     0     0     0     0     0     0
% no conditional heteroscedasticity
% our model captures the heteroscedasticity present in data


%t-distributed GARCH models
%Changing our definition of Mdl slightly
Mdl2=Mdl;% Mdl.Distribution = "Gaussian"
Mdl2.Distribution='t';
%Estimate the model
[estmdl2, EstParamCov2, logL2, info2]=estimate(Mdl2,retOtC);

% residuals for ARMA(1,1)-GARCH(1,1) model (t-distributed)
[E2,V2,logl2]=infer(estmdl2,retOtC);

%Compute fitted standardized error term
z2=E2./sqrt(V2);
```

```
%We need to make sure that our specification using the t-
distribution is
%correct. We can use a Q-Q plot to check it.
%Firstly, we get the estimated DoF and define a t-distribution with
the
%estimated DoF
dof=estmdl2.Distribution.DoF; % 8.3607
dist=makedist('tLocationScale',0,1,dof);

%We can then plot the standardized residual accordingly
% to compare the distribution of our standardized residuals and
% the t distribution
qqplot(z2,dist);
%It is clear that the goodness-of-fit is much better compared to a
normal
%assumption.

%Asymmetric GARCH. We consider only EGARCH
mdl3=arima('ARLags',1,'MALags',1,'Variance',egarch(1,1));
[estmdl3, EstParamCov3,logL3, info3]=estimate(mdl3,retOtC);

[E3,V3,logL3]=infer(estmdl3,ret); % residuals EGARHC(1,1)

%Compute fitted standardized error term
z3=E3./sqrt(V3);
% assessing EGARCH model
qqplot(z3);

%GJR-GARCH:
mdl4=arima('ARLags',1,'MALags',1,'Variance',gjr(1,1));
[estmdl4,EstParamCov4,logL4,info4] =estimate(mdl4,retOtC);

[E4,V4,logl4]=infer(estmdl4,retOtC);% residuals
z4=E4./sqrt(V4); % standardized residuals
qqplot(z4);

% GJR-GARCH specified with a t-distirbuted error term.
mdl6 = mdl4;
mdl6.Distribution='t';
[estmdl6, EstParamCov6, logL6, info6]=estimate(mdl6,retOtC);


 %   ARIMA(1,0,1) Model (t Distribution):

 %                 Value       StandardError    TStatistic
PValue
 %              ___________    _____________    ___________
___________
%
%    Constant    0.00013195      5.956e-05        2.2155
0.026729
%    AR{1}          0.87059       0.051299        16.971
0
%    MA{1}         -0.84175       0.057336       -14.681
0
```

```
 %    DoF             9.1731         1.6395         5.595
2.2063e-08


%    GJR(1,1) Conditional Variance Model (t Distribution):

 %                     Value       StandardError    TStatistic
PValue
 %                  __________    _____________    __________
__________

 %    Constant       2.5798e-06     8.5835e-07        3.0055
0.0026511
 %    GARCH{1}          0.89248       0.014831        60.178
0
 %    ARCH{1}          0.032586       0.013487         2.416
0.015691
  %   Leverage{1}      0.099503       0.020197        4.9267
8.3639e-07
  %   DoF                9.1731         1.6395         5.595
2.2063e-08

[E6,V6,logl6]=infer(estmdl6,retOtC);
z6=E6./sqrt(V6); % standardized residuals

dof=estmdl6.Distribution.DoF;
dist=makedist('tLocationScale',0,1,dof);
qqplot(z6,dist);

%Since the GJR-GARCH model nests the GARCH model, we can conduct a
%likelihood-ratio test to see if the extra parameter in the GJR-
GARCH model
%improves the likelihood significantly.
% first: compare ARMA(1,1)-GARCH(1,1) and ARMA(1,1)-GJR(1,1)
 % [h3,p3,stat3,cv3] = lratiotest(logL4,logl,1);
 % logl4:  unrestricted model
 % logl: restricted model
 % h3 = 1; p3 = 1.8839e-09
 % h3 = 1 means unrestricted model (GJR-GARCH) is better
%The test is significant at any conventional significance level,
suggesting
%that we should use an asymmetric structure when modelling financial
%returns with GARCH-type structure.

% 2nd: likelihood ratio test comparing ARMA(1,1)-GARCH(1,1) and
ARMA(1,1)_GJR(1,1) (t-distributed)
[h4,p4,stat4,c4] = lratiotest(logl6,logl,1);
% h4 : 1
% p4:  0

% Jarque Bera test for the ARMA(1,1)-GJR(1,1) (t-distributed)
[h7,p7,stat7,cv7]=jbtest(z6); % testing standardized residuals
% h7: 1 => not normally distributed
% p7: 1.0000e-03
```

```
%Detecting remaining ARCH effects for the ARMA(1,1)-GJR(1,1) (t-
distributed)
[h8,p8,stat8,cv8] = archtest(z6,'Lags',1:10);
% h8: 0   0   0   0   0   0   0   0   0   0 => no conditional
% heteroscedasticity
% our model captures the heteroscedasticity present in data

% Residual analysis for for the ARMA(1,1)-GJR(1,1) (t-distributed)
plotsize=[100,100,1200,550];
fig1 = figure;
hold on;
lag=100;
[acf1,~,bound]=autocorr(E6,lag); % 101 x 1 (from la 0 to lag 100)
[acf2,~,bound1]=autocorr(E6.^2,lag); % bound and bound1 are the same
p1=stem(1:1:lag,acf1(2:lag+1),'filled'); % ploting acf in blue and
filled
p2=stem(1:1:lag,acf2(2:lag+1)); %ploting acf^2 not filled in red
p=plot(1:1:lag,[ones(lag,1).*bound1(1) ones(lag,1).*bound1(2)]);
%ploting confidence bounds
hold off;
%Graph settings
p1.Color='r';
p1.MarkerSize=4;
p2.Color='b';
p2.MarkerSize=4;
p(1).LineWidth=2;
p(1).LineStyle=':';
p(1).Color='k';
p(2).LineWidth=2;
p(2).LineStyle=':';
p(2).Color='k';
title('Correlogram of the residual and squared residual');
legend('Residual', 'Squared residual','95% Confidence Bounds',
'Location','southoutside','Orientation','horizontal');


% Part 2: Realized Volatility signature plot

% e) Construct a volatility signature plot with daily annualized
realized
% volatility estimators based on the high-frequency data for the
chosen stock.
% cd('C:\Users\Eier\Documents\Lancaster\ACF609\Lab6')
% right click on mat folder in Lab6 (ACF609) add to path (folders +
subfolders)

%Set the time grid to compute the volatility signature plot
 tgrid=1:5:1800; % plot volatility signature plot = sampling
intervals in sec

 %Initialize a matrix for  volalatility signature plot
 VSP=zeros(length(dc),length(tgrid)); % store volatility signature
plo

 % 252 is the nr of trading days in a year
```

```
 %initialize various measures of RV
 rv=zeros(length(dc),1); % 252 x 1 matrise med elementer 0


%For loop for all trading days
for i=1:length(dc)
   pricet=data(data(:,1)==dc(i),3:4);

%Using the MFEtoolbox to construct volatility signature plot based
on the
%sampling frequencies in the tgrid
for j=1:length(tgrid)
   RV=realized_variance(pricet(:,2),pricet(:,1),...
       'seconds','CalendarTime',tgrid(j));
   VSP(i,j)=RV*252;
end
end

% f) Select an optimal sampling frequency and construct the daily
% annualized realized volatility estimator based on the optimal
sampling
% frequency.

%For loop for all trading days
for i=1:length(dc)
   pricet=data(data(:,1)==dc(i),3:4);

%Compute the 700-sec sample RV and
[RV,~]=realized_variance(pricet(:,2),pricet(:,1),...
       'seconds','CalendarTime',700,5); % split into 5 groups (k:
60)
                                        % 5 is nr of subsamples
                                        % 300 / 60 = 5 min !
                                        % every 300 min = 5 min
rv(i)=RV*252;%700 sec simple RV for day i (annualized)

end

figure('units','normalized','outerposition',[0 0 1 1]);
%Plot the volatility signature plot
plot(tgrid,mean(VSP)); % VSP
title('Volatility Signature Plot');
xlabel('Sampling Interval In Seconds');
ylabel('Average RV');

figure('units','normalized','outerposition',[0 0 1 1]);
%Plot RV measure
plot(rv);
legend({'700 sec RV'});

% g) Split the dataset into an in-sample period(03/01/2005 to
31/12/2012)
% and an out-of-sample period (02/01/2013 to 31/12/2014). Compute
the
% rolling-window , one-step-ahead (one-day-ahead) volatility
forecasts with
```

```
% the ARMA-GARCH model and a HAR-RV specification. To reduce
computational
% burden, estimate the ARMA-GARCH and HAR-RV model parameters only
once on
% the basis of the initial in-sample period.

T=length(retOtC);
%Set an insample period
insmpl=data(year(dateTbT_dt)<=2012,1:4);
%Similarly, set an out-of-sample period
outofsmpl=data(year(dateTbT_dt)>2012,1:4);

% compute open to close log returns for insample and out of sample
dcIn=unique(insmpl(:,1));
dcOut = unique(outofsmpl(:,1));

retOtCIn=zeros(length(dcIn),1);
retOtCOut=zeros(length(dcOut),1);

for i=1:length(dcIn)
   pricet=insmpl(insmpl(:,1)==dcIn(i),3:4);
   retOtCIn(i)=log(pricet(end,2))-log(pricet(1,2));

end

for i=1:length(dcOut)
   pricet=outofsmpl(outofsmpl(:,1)==dcOut(i),3:4);
   retOtCOut(i)=log(pricet(end,2))-log(pricet(1,2));
end

% cd('C:\Users\Eier\Documents\Lancaster\ACF609\Lab4');

% optimal model: ARMA(1,1)-GJR(1,1) (t-distributed)
mdl=arima('ARLags',1,'MALags',1,'Variance',gjr(1,1));
mdl2 = mdl;
mdl2.Distribution='t';
[estmdl, EstParamCov, logL, info]=estimate(mdl2,retOtC);
%Get the names of the series
RVnames = {'RV','GJR-GARCH-t'};

% [y, ymse, v ] = rolling_window_forecast( mdl,insmpl,outofsmpl,0 );
%We then  do rolling-window forecasts using different r.h.s.
variables.
%                                                    right hand
side
[E,V,logL]=infer(estmdl,retOtC);
%We store all the variables in a matrix below
RVmat = [rv , V];

%Choose the length of the insample period
% insmpl=504;  2013 + 504 = 2517
inLen = length(dcIn);
%Initialize a metrix to collect the forecasts:
res=[];
%Write a loop to compute the forecasts:
for i=1:size(RVmat,2)% i = 1,2,...9
```

```
   res=[res HAR_frcst( rv,RVmat(:,i),inLen )];
end
%Note that for the matrix res, all the odd columns store the true
value
%RV, and all the even colums store the forecasts from different
models

% h) For the out-of-sample period use the ex-post daily annualized
RV as a
% proxy of the forcasting target (true volatility) and compare the
% forecasting performances of the ARMA-GARCH and the HAR-RV models.
% You should compute standard forecasting statistics, conduct
appropriate
% tests and provide conclusions.

%Now to evaluate our forecasting performance for each estimator, we
can
%construct MSPE, MAPE and QLIKE measures on the big res matrix:
HAR_result=HAR_eval(res);
%Store the results in a nice table
HARtable=array2table(HAR_result,
'RowNames',RVnames,'VariableNames',...
    {'MAPE','MSPE','QLIKE'});
disp(HARtable);
```

```
    %                  MAPE          MSPE         QLIKE
    %                _________    __________    _______

     % RV             0.0086515    0.00015263    0.30441
     % GJR-GARCH-t     0.011787    0.00027487     0.5434
```

```
%Now, we can use a modified DM test to test if one model
significantly
%outperforms another. We will be comparing HAR-RV against GJR-
GARCH(1,1)
% To perform the modified DM test, we firstly
%compute the forecasting errors:
evec=res(:,2:2:end)-res(:,1:2:end);
%evec contains the forecasting error computed by forecast-true for
each
%series. Note that the 1st series is the RV benchmark.

%Initialize a matrix to store all results
DMresult = zeros(1,2);
 [DM,p_value] = dmtest_modified(evec(:,1), evec(:,2));
 DMresult(1,:)=[DM p_value];
% end
DMtbl=array2table(DMresult,'RowNames',{'GARCH vs
RV'},'VariableNames',{'TestStat','p_value'});
disp(DMtbl);
```

```
 %                  TestStat     p_value
 %                  ________    __________

  %  GARCH vs RV    -6.0161     3.4415e-09
```

Below follow functions provided in the labs used to solve this coursework.

```
function out=ADF_auto(data,dif,max_lags,sel_method)
% dfout=ADF_auto(retOtC,0,30,sel_method);
%This function computes the augmented Dickey Fuller (ADF test with
automatic lag
%selection based on an information criteria from AIC, BIC and HQIC.
Three
%sets of tests are performed: the ADF test with no drift or trend,
with a
%drift and with both drift and trend.

%Inputs:
%data: T-by-1 matrix of data => must be a vector


%dif: non-negative integer, the ADF test is performed after
differencing
%(0)
%the original series 'dif' times. (30)
%max_lags: non-negative integer, the maximum lags included in the
ADF test
%sel_method = selection_method: choose from 'AIC', 'BIC' and 'HQIC',
the information
%criteria minimized to select an optimal lag length

%output: A 6-by-3 table documenting
% the test statistics
% the critical values (corresponding to respective significance
level)
% p-value and
%the optimal lags chosen


%Checking selection methods
sel_method_choice={'AIC' 'BIC' 'HQIC'};

% if sel_method not i sel_method_choice (ismember() == 0)
if ismember(sel_method,sel_method_choice)==0
    % ismember(x,y)
    % returnerer 1 hvis el x er i liste y
    % 0 ellers
    error('Selection method not supported.');
end
%Checking input type of lags (must be integers)
% rem(a,b) returns remainder after division of a by b (integer
division)
% 0.1/1 = 0*1 + 0.1 so rem(0.1,1) = 0.1
% ~= is not equal to
if rem(dif,1)~=0 || rem(max_lags,1)~=0
      error('Wrong input type.');
end
%Differencing the data
sdata=data; % sdata er nå en kopi av data
while dif>0 % så lenge diff er større enn 0
sdata=diff(data,dif); % sdata is always equal to 1st difference ?
```

```
dif=dif-1;
end

warning('off');
%Do an ADF test for 0:max_lags with no drift or trend
[h,pValue,cValue,stat, regar]=adftest(sdata,'Model','AR', 'lags',
0:max_lags);
% Y = sdata = univariate time series
% 'alpha' significance level 0.05
% 'lags': nr of lagged difference terms to include in the model

%Do an ADF test for 0:max_lags with drift but no trend
% here I will obtained values for each lag
[hd,pValued,cValued,statd,regard] =
adftest(sdata,'Model','ARD','lags',0:max_lags);
% hd: 1 x 31
% pValued: 1 x 31
% cValued: 1 x 31
% statd: 1 x 31
% regard: see matlab (multi dimensional)
%Do an ADF test for 0:max_lags with drift and trend
[hts,pValuedts,cValuedts,statts,regts] =
adftest(sdata,'Model','TS','lags',0:max_lags);
%Store the information criteria in three different matrices
ICAR=[(regar.AIC); (regar.BIC); (regar.HQC)]; % 3 x 31, row =
selection method
ICARD=[(regard.AIC); (regard.BIC); (regard.HQC)];
ICTS=[(regts.AIC); (regts.BIC); (regts.HQC)];
% Choosing the minimum information criteria from the corresponding
matrix
[~, minICARi]=min(ICAR,[],2);
              % is a 3 x 1 vector containing the index of the
              % min value of each row (3
              % rows) with respect to which method is used
              % AIC, BIC or HQC
[~, minICARDi]=min(ICARD,[],2);
[~,minICTSi]=min(ICTS,[],2);
%Initialize the matrix Qtable to store results
Qtable=zeros(3,6);
%Choose the corresponding best lag according to the selected
information
%criterion

% tf = strcmp(s1,s2) compares s1 and s2 and returns 1 (true) if the
two are identical and 0 (false) otherwise.
% Text is considered identical if the size and content of each are
the same. The return result tf is of data type logical.
if strcmp(sel_method,'AIC')==1 % sel_method = 'AIC'
    lagar=minICARi(1);
    lagard=minICARDi(1);
    lagts=minICTSi(1);
elseif  strcmp(sel_method,'BIC')==1
    lagar=minICARi(2); % 11
    lagard=minICARDi(2); % 11
    lagts=minICTSi(2); % 11
else
```

```
    lagar=minICARi(3);
    lagard=minICARDi(3);
    lagts=minICTSi(3);
end
% Use the chosen lag to produce ADF test results
[h1,p1,df1,cv1,reg1] = adftest(sdata,'model','AR','alpha',[0.1 0.05
0.01],'lags',lagar);
[h2,p2,df2,cv2,reg2] = adftest(sdata,'model','ARD','alpha',[0.1 0.05
0.01],'lags',lagard);
[h3,p3,df3,cv3,reg3] = adftest(sdata,'model','TS','alpha',[0.1 0.05
0.01],'lags',lagts);
%Store the relevant statistics in a table
% 1st row, 1st col
Qtable(1,1)=df1(1); % test statistic as standard test statistic
% 1st row and 2 col
Qtable(2,1)=df2(1);
Qtable(3,1)=df3(1);
% 1st column, row 2-4
Qtable(1,2:4)=cv1; % critical values
% 2nd col, row 2-4
Qtable(2,2:4)=cv2; % critical values
% 3rd col, row 2-4
Qtable(3,2:4)=cv3; % critical values
% 1st col, 5th row
Qtable(1,5)=p1(1); % p value with respect to standard test statistic
% 2nd col, 5th row
Qtable(2,5)=p2(1);
% 3rd col, 5th row
Qtable(3,5)=p3(1);
% 1st col, 6th row
Qtable(1,6)=lagar-1; % appropriate nr of lags
% 2nd col, 6th row
Qtable(2,6)=lagard-1;
% 3rd col, 6th row
Qtable(3,6)=lagts-1;
%Prepare the output.
% Here: Qtable is transposed!!!
out=array2table(Qtable','RowNames', {'Test Stat' ' 10% C.V.' '5%
C.V.' '1% C.V.' 'P-value' 'Lag used'},...
'VariableNames', {'None' 'Drift' 'Trend'});
End

function [pbest,qbest ] = ARMA_optimal( data, pmax, qmax, method )
%This function computes the optimal AR and MA lags for an ARMA(p,q)
model
%based on information criteria. Inputs:
% data: T-by-1 vector of data to be modeled by an ARMA(p,q) model
% pmax: integer. Maximum AR lags considered in the model
% qmax: integer. Maximum MA lags considered in the model
% method: string. Methods to used for choosing the optimal model.
Choice:
%'AIC': Akaike Information Criterion
%'BIC': Bayesian Information Criterion
%'HQIC': Hannan-Quinn Information Criterion
%Outputs:
%pbest: optimal AR lags
```

```
%qbest: optimal MA lags.

T=length(data);
%Restrict the possible number of parameters:
if pmax+1+qmax>=T/10
    error('Too many parameters. Reduce pmax and qmax');
end

ICmat=zeros(pmax+1,qmax+1);


for i=1:pmax+1
    for j=1:qmax+1
        Mdltemp=arima(i-1,0,j-1);
       [~,~,logL,~] = estimate(Mdltemp,data,'Display','off');
       if strcmp(method,'AIC')
       ICmat(i,j)=-2*logL/T+2*(-1+i+j)/T;
       elseif strcmp(method,'BIC')
       ICmat(i,j)=-2*logL/T+(-1+i+j)*log(T)/T;
       elseif strcmp(method,'HQIC')
       ICmat(i,j)=-2*logL/T+2*(-1+i+j)*log(log(T))/T;
       else
           error('Unknown choice of information criterion');
       end
    end
end
%Find the location of the minimum information
[pbest,qbest]=find(ICmat==min(min(ICmat)));
pbest=pbest-1;
qbest=qbest-1;

end

function [gbest,abest,ICmat ] = GARCH_optimal( data, gmax, amax,
arp, map, method)
%This function computes the optimal AR and MA lags for an ARMA(p,q)
model
%based on information criteria. Inputs:
% data: T-by-1 vector of data to be modeled by an ARMA(p,q) model
(log ret)
% gmax: integer. Maximum GARCH lags considered in the model
% amax: integer. Maximum ARCH lags considered in the model
% arpar: number of AR parameters in the mean equation
% mapar: number of MA parameters in the mean equation
% method: string. Methods to used for choosing the optimal model.
Choice:
%'AIC': Akaike Information Criterion
%'BIC': Bayesian Information Criterion
%'HQIC': Hannan-Quinn Information Criterion
%Outputs:
%gbest: optimal GARCH lags
%abest: optimal ARCH lags.
%ICmat: Matrix of the chosen information criteria
```

```
T=length(data);
%Restrict the possible number of parameters:
% amax = maximum ARCH lags considered in the model
% gmax = maximum GARCH lags considered in the model
if amax+1+gmax>=T/10 % = 2517/10 = 251.7
    % Here amax + 1 g gmax = 2+1+2 = 5 < 251.7
    error('Too many parameters. Reduce pmax and qmax');
end

ICmat=zeros(1+gmax,amax); % (2 + 1) x 2 = 3 x 2 matrix

%start from GARCH(0,0), which is an ARMA(arp, arq) model). Note that
at
%least an ARCH term is required for a GARCH model.
for i=1:gmax+1 % i = 1,2,3
    for j=1:amax+1 % j = 1,2,3
        %If it is GARCH(0,0), treat it as an ARMA model
        if i*j==1 % if i = 1, and j = 1
                  % equivalent with GARCH(0,0)
            Mdltemp=arima('ARLags',arp, 'MAlags', map); %
ARIMA(arp,map)
            [EstMdl,EstParamCov,logL,info] =
estimate(Mdltemp,data,'Display','off');
        elseif i>1
            if j>1 % if not a GARCH(0,0) model = ARIMA(arp,map)
                Mdltemp=arima('ARLags',arp, 'MAlags',
map,'Variance',garch(i-1,j-1));
                [EstMdl,EstParamCov,logL,info] =
estimate(Mdltemp,data,'Display','off');
            else
                %If it is a GARCH(i,0) model, return negative
infinity as
                %likelihood
                logL=-Inf;
            end
        end

        if strcmp(method,'AIC')
            ICmat(i,j)=-2*logL/T+2*(arp+map+i+j)/T;
        elseif strcmp(method,'BIC')
            ICmat(i,j)=-2*logL/T+(arp+map+i+j)*log(T)/T;
        elseif strcmp(method,'HQIC')
            ICmat(i,j)=-2*logL/T+2*(arp+map+i+j)*log(log(T))/T;
        else
            error('Unknown choice of information criterion');
        end
    end
end
%Find the location of the minimum information
[gbest,abest]=find(ICmat==min(min(ICmat)));
gbest=gbest-1;
abest=abest-1;

end
```

```
function [DM,p_value] = dmtest_modified(e1, e2, h)
%DMTEST: Retrieves the Diebold-Mariano test statistic (1995) for the
% equality of forecast accuracy of two forecasts under general
assumptions.
%
%   DM = dmtest(e1, e2, ...) calculates the D-M test statistic on
the base
%   of the loss differential which is defined as the difference of
the
%   squared forecast errors
%
%   In particular, with the DM statistic one can test the null
hypothesis:
%   H0: E(d) = 0. The Diebold-Mariano test assumes that the loss
%   differential process 'd' is stationary and defines the statistic
as:
%   DM = mean(d) / sqrt[ (1/T) * VAR(d) ]  ~ N(0,1),
%   where VAR(d) is an estimate of the unconditional variance of
'd'.
%
%   This function also corrects for the autocorrelation that multi-
period
%   forecast errors usually exhibit. Note that an efficient h-period
%   forecast will have forecast errors following MA(h-1) processes.
%   Diebold-Mariano use a Newey-West type estimator for sample
variance of
%   the loss differential to account for this concern.
%
%   'e1' is a 'T1-by-1' vector of the forecast errors from the first
model
%   'e2' is a 'T2-by-1' vector of the forecast errors from the
second model
%
%   It should hold that T1 = T2 = T.
%
%   DM = DMTEST(e1, e2, 'h') allows you to specify an additional
parameter
%   value 'h' to account for the autocorrelation in the loss
differential
%   for multi-period ahead forecasts.
%       'h'         the forecast horizon, initially set equal to 1
%
%   DM = DMTEST(...) returns a constant:
%       'DM'      the Diebold-Mariano (1995) test statistic
%
%  Semin Ibisevic (2011)
%  $Date: 11/29/2011 $
%
% ----------------------------------------------------------------
-------
% References
% K. Bouman. Quantitative methods in international finance and
% macroeconomics. Econometric Institute, 2011. Lecture FEM21004-11.
%
% Diebold, F.X. and R.S. Mariano (1995), "Comparing predictive
accuracy",
```

```
% Journal of Business & Economic Statistics, 13, 253-263.
% -----------------------------------------------------------------
-------

if nargin < 2
    error('dmtest:TooFewInputs','At least two arguments are
required');
end
if nargin < 3
   h = 1;
end
if size(e1,1) ~= size(e2,1) || size(e1,2) ~= size(e2,2)
    error('dmtest:InvalidInput','Vectors should be of equal
length');
end
if size(e1,2) > 1 || size(e2,2) > 1
    error('dmtest:InvalidInput','Input should have T rows and 1
column');
end

% Initialization
T = size(e1,1);

% Define the loss differential
d = abs(e1).^2 - abs(e2).^2;

% Ralculate the variance of the loss differential, taking into
account
% autocorrelation.
dMean = mean(d);
gamma0 = var(d);
if h > 1
    gamma = zeros(h-1,1);
    for i = 1:h-1
        sampleCov = cov(d(1+i:T),d(1:T-i),1);
        gamma(i) = sampleCov(2);
    end
    varD = gamma0 + 2*sum(gamma);
else
    varD = gamma0;
end
%k is calculated to adjuste the statistic as per Harvey, Leybourne,
and Newbold (1997)
k = ((T+1-2*h+(((h)*(h-1))/T))/T)^(1/2);
% Retrieve the diebold mariano statistic DM ~N(0,1)
    DM = (dMean / sqrt ( (varD/T) ))*k;
%P_VALUE is calculated
    p_value = 2*tcdf(-abs(DM),T-1)
end



clearvars; clc;
%Load the dataset of RV measures
cd('C:\Users\Eier\Documents\Lancaster\ACF609\Lab7')
```

```
load('RVtbl.mat');
% RVtble: 2517 x 10
%Plot the RV measures.
%plot(RVtbl{:,2:10}); Fig 1

%Get the names of the series
RVnames=RVtbl.Properties.VariableNames(2:10);
%We choose the benchmark model to be RK as the l.h.s. of a HAR model
%                                                left hand side
% {'tickRV'}    {'RV'}    {'SubRV'}    {'TSRV'}    {'RK'}
{'preRV'}    {'RBV'}    {'preRBV'}    {'GARCH'}
RK=RVtbl.RK; % 2517 x 1

%We then  do rolling-window forecasts using different r.h.s.
variables.
%                                                       right hand
side
%We store all the variables in a matrix below
RVmat=RVtbl{:,2:10}; % 2517 x 9


%Choose the length of the insample period
insmpl=504; % 2013 + 504 = 2517
%Initialize a metrix to collect the forecasts:
res=[];
%Write a loop to compute the forecasts:
for i=1:size(RVmat,2)% i = 1,2,...9
   res=[res HAR_frcst( RK,RVmat(:,i),insmpl )]; % size: 2013 x 18
   %     res er hva vi har lagret før i res
end
% res
%Note that for the matrix res, all the odd columns store the true
value
%RK, and all the even colums store the forecasts from different
models, in
%the same order as stored in the RVtbl.mat

%Now to evaluate our forecasting performance for each estimator, we
can
%construct MSPE, MAPE and QLIKE measures on the big res matrix:
HAR_result=HAR_eval(res);% 9 x 3 matrise
%Store the results in a nice table
HARtable=array2table(HAR_result,
'RowNames',RVnames,'VariableNames',...
    {'MAPE','MSPE','QLIKE'});
disp(HARtable);

 %                MAPE          MSPE         QLIKE
             _________    _________    _______

 %   tickRV    8.0917e-05    4.7462e-08    0.28455
 %   RV         8.029e-05    4.8377e-08    0.28304
 %   SubRV     8.0846e-05    4.7185e-08    0.28137
 %   TSRV      8.1885e-05    5.1172e-08     0.2944
 %   RK        7.9252e-05    4.4487e-08    0.27677 % less noisy
benchmark
```

```
 %    preRV     7.9483e-05    4.4965e-08    0.27553 %prerv qlike
lowest
 %   RBV       8.1834e-05    5.1018e-08    0.29021
 %   preRBV    7.9601e-05    4.5145e-08    0.28097
 %   GARCH     9.2406e-05    5.9163e-08    0.34501 % more dependence
 %   between days
 % IC : 2 IC says RK is best while preRV is best (IC:1)
%From the table we see that different loss functions lead to
differnt
%conclusions for the best model, so according to MAPE and MSPE, RK
performs the
%best, but QLIKE suggests that preRV performs the best. Also, all
three
%loss functions suggest that GARCH measure performs the worst, which
is not
%surprising because GARCH measure do not take intraday price
information
%into account

%Now, we can use a modified DM test to test if one model
significantly
%outperforms another. We will be comparing HAR-RK against all
possible
%alternatives in the dataset. To perform the modified DM test, we
firstly
%compute the forecasting errors:
evec=res(:,2:2:end)-res(:,1:2:end);
     % res[ kol2 kol4 ...kol8] - res[kol1 kol3 .. kol9] = forecasted
- true
% size(evec) 2013 x 9
% evec[5] = RK = true != 0
%evec contains the forecasting error computed by forecast-true for
each
%series. Note that the 5th series is the RK benchmark. To make our
%programme simpler, we rearrange the order of the collums and move
RK
%the first column of evec:
evec=evec(:,[5 1:4 6:9]); % flytter RK (true) til første kolonne

%Initialize a matrix to store all results
DMresult=zeros(8,2); % compare 8 RV values to RK (true)
for i=1:8
    %We are testing HAR-RK against all other alternatives. Therefore
the
    %first entries will always be the loss from HAR-RK
    % e1: RK
    % e2: andre 8 RV modeller
 [DM,p_value] = dmtest_modified(evec(:,1), evec(:,i+1));
 DMresult(i,:)=[DM p_value];
end
DMtbl=array2table(DMresult,'RowNames',{'tickRV vs RK'...
    'RV vs RK' 'SubRV vs RK' 'TSRV vs RK' 'preRV vs RK' 'RBV vs
RK'...
    'preRBV vs RK' 'GARCH vs
RK'},'VariableNames',{'TestStat','p_value'});
disp(DMtbl);
```

```
%From the table we can see that all TestStats are negative, which
suggests
%that the HAR-RK model outperform HAR models with other r.h.s.
variables.
%However this difference in terms of MSPE is not significantly
different
%from zero since the p-values are all very high, except from HAR-
GARCH as
%we see a clear rejection. This indicates that the HAR-GARCH model
performs
%significantly worse than HAR-RK in predicting RK with a much larger
%forecasting error.

%                    TestStat     p_value
%                    ________    _________

%    tickRV vs RK    -0.81038      0.41782  % not stat sign diff -
fail to
                                            % rekect that models are
stat
                                            % sign diff
 %   RV vs RK        -0.74277      0.45771
 %   SubRV vs RK     -0.86881      0.38506
%    TSRV vs RK      -0.95268      0.34087
%    preRV vs RK     -0.43457      0.66392
%    RBV vs RK        -1.0536      0.29217
%    preRBV vs RK    -0.45577       0.6486
%    GARCH vs RK      -2.8285    0.0047226   GARCH is worse
significantly
                                             %    diff not designed
                                             % for tick by tick data


function ret = HAR_eval(arg)
%This function computes the MAPE, MSPE and QLIKE of the forecasts
from
%different series.
%Input: arg is a (T-by-2*C) matrix of the following form:
%[true forecast1 true forecast2 true forecast3 ... true forecastC ]
%Output: ret is a (C-by-3) matrix:
%[MAPE1 MSPE1 QLIKE1; MAPE2 MSPE2 QLIKE2 ; ... ; MAPEC MSPEC QLIKEC]

%Get the number of columns in the arg
C=size(arg,2);
%Set an index for the forecasts that retrieves all the even columns
from
%arg
ind=2:2:C;
%Initialize the return series.
ret=zeros(C/2,3);
%Computing MAPE
ret(:,1)=mean(abs(arg(:,ind)-arg(:,ind-1)))';
%Computing MSPE
ret(:,2)=mean((arg(:,ind)-arg(:,ind-1)).^2)';
%Computing QLIKE
```

```
ret(:,3)=mean(arg(:,ind)./arg(:,ind-1)-log(arg(:,ind)./arg(:,ind-
1))-1)';
end


function ret= HAR_frcst( Y,X,insmpl )
%This function computes one-step ahead rolling window HAR forecasts
with
%re-estimation with the HAR model:
%Y(t)=w+b_d*X_d(t-1)+b_w*X_w(t-1)+b_m*X_m(t-1)+u(t)
%Inputs: Y: a T-by-1 vector used as the l.h.s. of the HAR model - RK
%        X: a T-by-1 vector used as the r.h.s. of the HAR model
%        insmpl: a integer indicating the size of the initial
estimation
%        window - 504
%Outputs: ret: a (T-insmpl)-by-2 vector. The first column is the
true value
%and the second column stores the forecasted value
% 2013 x 2

%Input Checking
if length(Y)~=length(X)
   error('Size mismatch between Y and X');
end
if insmpl>=length(Y)
   error('In-sample period should be shorter than the length of the
data');
end

%Extract the length of the dataset
T=length(Y); % 2517
%Get the number of re-estimations of the HAR model
N=T-insmpl; % = 2517 - 504 = 2013
%Initialize a vector to compute results
ret=zeros(N,2);% 2013 x 2
%Set the first column of ret to be true values
ret(:,1)=Y(insmpl+1:T);
% første kolonne i ret = Y[504 + 1:2517] = RK[505:2517] = RK[out of
sample values]
%
for i=1:N % 1,2,...,2013 (for each forecasted day)
    %Get temporary Y and X for each re-estimation of HAR. Think
about the
    %index i:insmpl+i-1!
    %rolling insmple values
    Ytemp=Y(i:insmpl+i-1); %
RK[1:504]=insmpl,RK[2:505],RK[3:506],...,RK[2013:2516] (all have
length: 504 = insmpl size)
    Xtemp=X(i:insmpl+i-1); % RV[1:504],RV[2:505],...,RV[2013:2516]
(all have length: 504, which is insmpl size)
    %Compute the r.h.s. and l.h.s. variables using the HARX.m
function
    [h,XF]= HARX( Ytemp,Xtemp);
    % h: insmpl x 4 matrix = 482 x 4 = [Y, X_daily, X_weekly,
X_monthly]
    % XF:  a 1-by-3 vector that can be used to forecast Y(t+1) %
```

```
    % coefficients

    %OLS regression by hand...
    %HAR
    if i == 1
         har=[ ones(insmpl-22,1) h(:,2:4)]; % 482 x 4  matrix
     % = [ones(504-22 = 482,1) [X_daily X_weekly X_monthly]
     % = [ har1 har2 har3 har4]
     % hvor har1 bare er en kolonne av lengde 482 med elementet 1

         b=(har'*har)\har'*h(:,1); % beta
     % = (transpose(har)*har) % transpose(har)* h(:,1)
         ret(i,2)=[1 XF]*b; % forecasted value 1 ...  2013
    else %                                        *    Y
     % b
     % 0.0000
    % 0.0258
    % 0.6119
   % -0.1246

     %Compute the one-step ahead forecast and store it
     ret(i,2)=[1 XF]*b; % forecasted value 1 ...  2013
    end
end
end



function [ret,XF]= HARX( Y,X )
%This function is used to compute the l.h.s. and r.h.s. variables
for a HAR
%model: Y(t)=w+b_d*X_d(t-1)+b_w*X_w(t-1)+b_m*X_m(t-1)+u(t)
%Input: Y: a T-by-1 vector for the l.h.s. of HAR model = RK, T = 504
=
                     %insmpl size so Y: 504 x 1 matrix, the insmple
rolling values
%       X: a T-by-1 vector for the r.h.s. of HAR model = RV which is
also
         % a 504 x 1 matrix, the insmple rolling values
%Output: ret: a (T-22)-by-4 matrix. [Y, X_d, X_w, X_m] => a (504-22)
x 22 matrix
%        = (482 x 22) matrix
%        XF:  a 1-by-3 vector that can be used to forecast Y(t+1)

%Input Checking
if length(Y)~=length(X) % must be of same size
   error('Size mismatch between Y and X');
end

%Number of rows in Y
r=length(Y); % length of insample rolling window = 504
%Initialize the return matrix for forecasts
ret=zeros(r-22,4); % size: (504-22) x 4 = 482 x 4
% ret = [Y X_d, X_w X_m]

%Retrieve the vector of Y as l.h.s. of HAR
```

```
ret(:,1)=Y(23:r); % first column in ret = forecasts
% RK[23:504] = y

%Retrieve the lagged vector of X as X_daily
ret(:,2)=X(22:r-1); % 2nd column
% X_d = RV(22:504-1) = RV(22:503)

%Computer weekly and monthly moving averages
Xw=movmean(X,[4,0]); % 504 x 1
Xm=movmean(X,[21,0]); % 504 x 1
%Store the corresponding X_w and X_m
ret(:,3)=Xw(22:r-1);% = Xw(22:503) - 3rd column => lengde 482
ret(:,4)=Xm(22:r-1);% = Xm(22:504) - 4rth column => lengde 482
%Store X_d(T) X_w(T) and X_m(T) as XF.
XF=[X(r) Xw(r) Xm(r)]; %  XF = [X(504) Xw(504) Xm(504)]
end

% XF = [X(r) Xw(r) Xm(r)]

% XF =

   % 1.0e-03 *

    % 0.1533    0.0564    0.0717
```