Appendices

Appendix 1: Matlab Code

Since the code applied to the European and American markets are very similar, we will here only present the one applied to the former.

1. Development of the ADCC model

```
%% Pre-processibg data
% First, import the MFE toolbox by Kevin Sheppard

pathtool

% Importing the price series of stock indexes and government bonds

EU_EQ=xlsread('/Users/romaindeharlez/Documents/EU data.xlsx','EU eq','B296:B3904');
EU_B=xlsread('/Users/romaindeharlez/Documents/EU data.xlsx','EU B','B296:B3904');

% Reading dates

Date=xlsread('/Users/romaindeharlez/Documents/EU data.xlsx','EU eq','A296:A3904');
Dates=x2mdate(Date);
fancy_dates=datestr(Dates(2:end,1));

% Computing returns

EU_EQR=tick2ret(EU_EQ,Dates,'Continuous');
EU_BR=tick2ret(EU_B,Dates,'Continuous');

% Demeaning the returns

mean_EU_EQR=mean(EU_EQR);
EU_EQR_C=EU_EQR-mean_EU_EQR;
mean_EU_BR=mean(EU_BR);
EU_BR_C=EU_BR-mean_EU_BR;
EU_C=[EU_EQR_C EU_BR_C];

%% Pre-estimation analysis

% Stationarity test (adf)

[hQ3,pQ3]=adftest(EU_EQR_C);
[hB3,pB3]=adftest(EU_BR_C);

% Normality tests

qqplot(EU_EQR_C)
[hQ,pQ]=jbtest(EU_EQR_C);
qqplot(EU_BR_C)
[hB,pB]=jbtest(EU_BR_C);

% Autocorrelation testing
```

```
[hQ1,pQ1]=lbqtest(power(EU_EQR_C,2));
[hB1,pB1]=lbqtest(power(EU_BR_C,2));

% ARCH effect testing

[hQ2,pQ2] = archtest(EU_EQR_C);
[hB2,pB2] = archtest(EU_BR_C);
[hQ2_5,pQ2_5] = archtest(EU_EQR_C,'lags',5);
[hB2_5,pB2_5] = archtest(EU_BR_C,'lags',5);
[hQ2_10,pQ2_10] = archtest(EU_EQR_C,'lags',10);
[hB2_10,pB2_10] = archtest(EU_BR_C,'lags',10);

%% testing the fit of the different Garch models

% For equities first

resultsE=zeros(20,8);

[paramE1011, LL1011, ~, VCVE1011, ~, ~, ~] = tarch(EU_EQR_C, 1, 0, 1, 'NORMAL', 1, [], []);
[paramE1111, LL1111, ~, VCVE1111, ~, ~, ~] = tarch(EU_EQR_C, 1, 1, 1, 'NORMAL', 1, [], []);
[paramE1121, LL1121, ~, VCVE1121, ~, ~, ~] = tarch(EU_EQR_C, 1, 1, 2, 'NORMAL', 1, [], []);
[paramE2111, LL2111, ~, VCVE2111, ~, ~, ~] = tarch(EU_EQR_C, 2, 1, 1, 'NORMAL', 1, [], []);
[paramE1211, LL1211, ~, VCVE1211, ~, ~, ~] = tarch(EU_EQR_C, 1, 2, 1, 'NORMAL', 1, [], []);

[paramE1012, LL1012, ~, VCVE1012, ~, ~, ~] = tarch(EU_EQR_C, 1, 0, 1, 'NORMAL', 2, [], []);
[paramE1112, LL1112, ~, VCVE1112, ~, ~, ~] = tarch(EU_EQR_C, 1, 1, 1, 'NORMAL', 2, [], []);
[paramE1122, LL1122, ~, VCVE1122, ~, ~, ~] = tarch(EU_EQR_C, 1, 1, 2, 'NORMAL', 2, [], []);
[paramE2112, LL2112, ~, VCVE2112, ~, ~, ~] = tarch(EU_EQR_C, 2, 1, 1, 'NORMAL', 2, [], []);
[paramE1212, LL1212, ~, VCVE1212, ~, ~, ~] = tarch(EU_EQR_C, 1, 2, 1, 'NORMAL', 2, [], []);

resultsE(1,8)=aicbic(LL1011,3);
resultsE(3,8)=aicbic(LL1111,3);
resultsE(5,8)=aicbic(LL1121,3);
resultsE(7,8)=aicbic(LL2111,3);
resultsE(9,8)=aicbic(LL1211,3);
resultsE(11,8)=aicbic(LL1012,3);
resultsE(13,8)=aicbic(LL1112,3);
resultsE(15,8)=aicbic(LL1122,3);
resultsE(17,8)=aicbic(LL2112,3);
resultsE(19,8)=aicbic(LL1212,3);


i=1;

resultsE(i,1)=paramE1011(1,1);
[~,resultsE(i+1,1)]=ztest(paramE1011(1,1),0,sqrt(VCVE1011(1,1)));
resultsE(i,2)=paramE1011(2,1);
[~,resultsE(i+1,2)]=ztest(paramE1011(2,1),0,sqrt(VCVE1011(2,2)));
resultsE(i,6)=paramE1011(3,1);
[~,resultsE(i+1,6)]=ztest(paramE1011(3,1),0,sqrt(VCVE1011(3,3)));

i=3;

resultsE(i,1)=paramE1111(1,1);
[~,resultsE(i+1,1)]=ztest(paramE1111(1,1),0,sqrt(VCVE1111(1,1)));
resultsE(i,2)=paramE1111(2,1);
[~,resultsE(i+1,2)]=ztest(paramE1111(2,1),0,sqrt(VCVE1111(2,2)));
resultsE(i,4)=paramE1111(3,1);
[~,resultsE(i+1,4)]=ztest(paramE1111(3,1),0,sqrt(VCVE1111(3,3)));
```

```
resultsE(i,6)=paramE1111(4,1);
[~,resultsE(i+1,6)]=ztest(paramE1111(4,1),0,sqrt(VCVE1111(4,4)));

i=5;

resultsE(i,1)=paramE1121(1,1);
[~,resultsE(i+1,1)]=ztest(paramE1121(1,1),0,sqrt(VCVE1121(1,1)));
resultsE(i,2)=paramE1121(2,1);
[~,resultsE(i+1,2)]=ztest(paramE1121(2,1),0,sqrt(VCVE1121(2,2)));
resultsE(i,4)=paramE1121(3,1);
[~,resultsE(i+1,4)]=ztest(paramE1121(3,1),0,sqrt(VCVE1121(3,3)));
resultsE(i,6)=paramE1121(4,1);
[~,resultsE(i+1,6)]=ztest(paramE1121(4,1),0,sqrt(VCVE1121(4,4)));
resultsE(i,7)=paramE1121(5,1);
[~,resultsE(i+1,7)]=ztest(paramE1121(5,1),0,sqrt(VCVE1121(5,5)));

i=7;

resultsE(i,1)=paramE2111(1,1);
[~,resultsE(i+1,1)]=ztest(paramE2111(1,1),0,sqrt(VCVE2111(1,1)));
resultsE(i,2)=paramE2111(2,1);
[~,resultsE(i+1,2)]=ztest(paramE2111(2,1),0,sqrt(VCVE2111(2,2)));
resultsE(i,3)=paramE2111(3,1);
[~,resultsE(i+1,3)]=ztest(paramE2111(3,1),0,sqrt(VCVE2111(3,3)));
resultsE(i,4)=paramE2111(4,1);
[~,resultsE(i+1,4)]=ztest(paramE2111(4,1),0,sqrt(VCVE2111(4,4)));
resultsE(i,6)=paramE2111(5,1);
[~,resultsE(i+1,6)]=ztest(paramE2111(5,1),0,sqrt(VCVE2111(5,5)));

i=9;

resultsE(i,1)=paramE1211(1,1);
[~,resultsE(i+1,1)]=ztest(paramE1211(1,1),0,sqrt(VCVE1211(1,1)));
resultsE(i,2)=paramE1211(2,1);
[~,resultsE(i+1,2)]=ztest(paramE1211(2,1),0,sqrt(VCVE1211(2,2)));
resultsE(i,4)=paramE1211(3,1);
[~,resultsE(i+1,4)]=ztest(paramE1211(3,1),0,sqrt(VCVE1211(3,3)));
resultsE(i,5)=paramE1211(4,1);
[~,resultsE(i+1,5)]=ztest(paramE1211(4,1),0,sqrt(VCVE1211(4,4)));
resultsE(i,6)=paramE1211(5,1);
[~,resultsE(i+1,6)]=ztest(paramE1211(5,1),0,sqrt(VCVE1211(5,5)));

i=11;

resultsE(i,1)=paramE1012(1,1);
[~,resultsE(i+1,1)]=ztest(paramE1012(1,1),0,sqrt(VCVE1012(1,1)));
resultsE(i,2)=paramE1012(2,1);
[~,resultsE(i+1,2)]=ztest(paramE1012(2,1),0,sqrt(VCVE1012(2,2)));
resultsE(i,6)=paramE1012(3,1);
[~,resultsE(i+1,6)]=ztest(paramE1012(3,1),0,sqrt(VCVE1012(3,3)));

i=13;

resultsE(i,1)=paramE1112(1,1);
[~,resultsE(i+1,1)]=ztest(paramE1112(1,1),0,sqrt(VCVE1112(1,1)));
resultsE(i,2)=paramE1112(2,1);
[~,resultsE(i+1,2)]=ztest(paramE1112(2,1),0,sqrt(VCVE1112(2,2)));
resultsE(i,4)=paramE1112(3,1);
[~,resultsE(i+1,4)]=ztest(paramE1112(3,1),0,sqrt(VCVE1112(3,3)));
resultsE(i,6)=paramE1112(4,1);
```

```
[~,resultsE(i+1,6)]=ztest(paramE1112(4,1),0,sqrt(VCVE1112(4,4)));

i=15;

resultsE(i,1)=paramE1122(1,1);
[~,resultsE(i+1,1)]=ztest(paramE1122(1,1),0,sqrt(VCVE1122(1,1)));
resultsE(i,2)=paramE1122(2,1);
[~,resultsE(i+1,2)]=ztest(paramE1122(2,1),0,sqrt(VCVE1122(2,2)));
resultsE(i,4)=paramE1122(3,1);
[~,resultsE(i+1,4)]=ztest(paramE1122(3,1),0,sqrt(VCVE1122(3,3)));
resultsE(i,6)=paramE1122(4,1);
[~,resultsE(i+1,6)]=ztest(paramE1122(4,1),0,sqrt(VCVE1122(4,4)));
resultsE(i,7)=paramE1122(5,1);
[~,resultsE(i+1,7)]=ztest(paramE1122(5,1),0,sqrt(VCVE1122(5,5)));

i=17;

resultsE(i,1)=paramE2112(1,1);
[~,resultsE(i+1,1)]=ztest(paramE2112(1,1),0,sqrt(VCVE2112(1,1)));
resultsE(i,2)=paramE2112(2,1);
[~,resultsE(i+1,2)]=ztest(paramE2112(2,1),0,sqrt(VCVE2112(2,2)));
resultsE(i,3)=paramE2112(3,1);
[~,resultsE(i+1,3)]=ztest(paramE2112(3,1),0,sqrt(VCVE2112(3,3)));
resultsE(i,4)=paramE2112(4,1);
[~,resultsE(i+1,4)]=ztest(paramE2112(4,1),0,sqrt(VCVE2112(4,4)));
resultsE(i,6)=paramE2112(5,1);
[~,resultsE(i+1,6)]=ztest(paramE2112(5,1),0,sqrt(VCVE2112(5,5)));

i=19;

resultsE(i,1)=paramE1212(1,1);
[~,resultsE(i+1,1)]=ztest(paramE1212(1,1),0,sqrt(VCVE1212(1,1)));
resultsE(i,2)=paramE1212(2,1);
[~,resultsE(i+1,2)]=ztest(paramE1212(2,1),0,sqrt(VCVE1212(2,2)));
resultsE(i,4)=paramE1212(3,1);
[~,resultsE(i+1,4)]=ztest(paramE1212(3,1),0,sqrt(VCVE1212(3,3)));
resultsE(i,5)=paramE1212(4,1);
[~,resultsE(i+1,5)]=ztest(paramE1212(4,1),0,sqrt(VCVE1212(4,4)));
resultsE(i,6)=paramE1212(5,1);
[~,resultsE(i+1,6)]=ztest(paramE1212(5,1),0,sqrt(VCVE1212(5,5)));

% For bonds now

resultsB=zeros(20,8);

[paramB1011, LLB1011, ~, VCVB1011, ~, ~, ~] = tarch(EU_BR_C, 1, 0, 1, 'NORMAL', 1, [], []);
[paramB1111, LLB1111, ~, VCVB1111, ~, ~, ~] = tarch(EU_BR_C, 1, 1, 1, 'NORMAL', 1, [], []);
[paramB1121, LLB1121, ~, VCVB1121, ~, ~, ~] = tarch(EU_BR_C, 1, 1, 2, 'NORMAL', 1, [], []);
[paramB2111, LLB2111, ~, VCVB2111, ~, ~, ~] = tarch(EU_BR_C, 2, 1, 1, 'NORMAL', 1, [], []);
[paramB1211, LLB1211, ~, VCVB1211, ~, ~, ~] = tarch(EU_BR_C, 1, 2, 1, 'NORMAL', 1, [], []);

[paramB1012, LLB1012, ~, VCVB1012, ~, ~, ~] = tarch(EU_BR_C, 1, 0, 1, 'NORMAL', 2, [], []);
[paramB1112, LLB1112, ~, VCVB1112, ~, ~, ~] = tarch(EU_BR_C, 1, 1, 1, 'NORMAL', 2, [], []);
[paramB1122, LLB1122, ~, VCVB1122, ~, ~, ~] = tarch(EU_BR_C, 1, 1, 2, 'NORMAL', 2, [], []);
[paramB2112, LLB2112, ~, VCVB2112, ~, ~, ~] = tarch(EU_BR_C, 2, 1, 1, 'NORMAL', 2, [], []);
[paramB1212, LLB1212, ~, VCVB1212, ~, ~, ~] = tarch(EU_BR_C, 1, 2, 1, 'NORMAL', 2, [], []);

resultsB(1,8)=aicbic(LLB1011,3);
resultsB(3,8)=aicbic(LLB1111,3);
resultsB(5,8)=aicbic(LLB1121,3);
```

```
resultsB(7,8)=aicbic(LLB2111,3);
resultsB(9,8)=aicbic(LLB1211,3);
resultsB(11,8)=aicbic(LLB1012,3);
resultsB(13,8)=aicbic(LLB1112,3);
resultsB(15,8)=aicbic(LLB1122,3);
resultsB(17,8)=aicbic(LLB2112,3);
resultsB(19,8)=aicbic(LLB1212,3);


i=1;

resultsB(i,1)=paramB1011(1,1);
[~,resultsB(i+1,1)]=ztest(paramB1011(1,1),0,sqrt(VCVB1011(1,1)));
resultsB(i,2)=paramB1011(2,1);
[~,resultsB(i+1,2)]=ztest(paramB1011(2,1),0,sqrt(VCVB1011(2,2)));
resultsB(i,6)=paramB1011(3,1);
[~,resultsB(i+1,6)]=ztest(paramB1011(3,1),0,sqrt(VCVB1011(3,3)));

i=3;

resultsB(i,1)=paramB1111(1,1);
[~,resultsB(i+1,1)]=ztest(paramB1111(1,1),0,sqrt(VCVB1111(1,1)));
resultsB(i,2)=paramB1111(2,1);
[~,resultsB(i+1,2)]=ztest(paramB1111(2,1),0,sqrt(VCVB1111(2,2)));
resultsB(i,4)=paramB1111(3,1);
[~,resultsB(i+1,4)]=ztest(paramB1111(3,1),0,sqrt(VCVB1111(3,3)));
resultsB(i,6)=paramB1111(4,1);
[~,resultsB(i+1,6)]=ztest(paramB1111(4,1),0,sqrt(VCVB1111(4,4)));

i=5;

resultsB(i,1)=paramB1121(1,1);
[~,resultsB(i+1,1)]=ztest(paramB1121(1,1),0,sqrt(VCVB1121(1,1)));
resultsB(i,2)=paramB1121(2,1);
[~,resultsB(i+1,2)]=ztest(paramB1121(2,1),0,sqrt(VCVB1121(2,2)));
resultsB(i,4)=paramB1121(3,1);
[~,resultsB(i+1,4)]=ztest(paramB1121(3,1),0,sqrt(VCVB1121(3,3)));
resultsB(i,6)=paramB1121(4,1);
[~,resultsB(i+1,6)]=ztest(paramB1121(4,1),0,sqrt(VCVB1121(4,4)));
resultsB(i,7)=paramB1121(5,1);
[~,resultsB(i+1,7)]=ztest(paramB1121(5,1),0,sqrt(VCVB1121(5,5)));

i=7;

resultsB(i,1)=paramB2111(1,1);
[~,resultsB(i+1,1)]=ztest(paramB2111(1,1),0,sqrt(VCVB2111(1,1)));
resultsB(i,2)=paramB2111(2,1);
[~,resultsB(i+1,2)]=ztest(paramB2111(2,1),0,sqrt(VCVB2111(2,2)));
resultsB(i,3)=paramB2111(3,1);
[~,resultsB(i+1,3)]=ztest(paramB2111(3,1),0,sqrt(VCVB2111(3,3)));
resultsB(i,4)=paramB2111(4,1);
[~,resultsB(i+1,4)]=ztest(paramB2111(4,1),0,sqrt(VCVB2111(4,4)));
resultsB(i,6)=paramB2111(5,1);
[~,resultsB(i+1,6)]=ztest(paramB2111(5,1),0,sqrt(VCVB2111(5,5)));

i=9;

resultsB(i,1)=paramB1211(1,1);
[~,resultsB(i+1,1)]=ztest(paramB1211(1,1),0,sqrt(VCVB1211(1,1)));
resultsB(i,2)=paramB1211(2,1);
```

```
[~,resultsB(i+1,2)]=ztest(paramB1211(2,1),0,sqrt(VCVB1211(2,2)));
resultsB(i,4)=paramB1211(3,1);
[~,resultsB(i+1,4)]=ztest(paramB1211(3,1),0,sqrt(VCVB1211(3,3)));
resultsB(i,5)=paramB1211(4,1);
[~,resultsB(i+1,5)]=ztest(paramB1211(4,1),0,sqrt(VCVB1211(4,4)));
resultsB(i,6)=paramB1211(5,1);
[~,resultsB(i+1,6)]=ztest(paramB1211(5,1),0,sqrt(VCVB1211(5,5)));

i=11;

resultsB(i,1)=paramB1012(1,1);
[~,resultsB(i+1,1)]=ztest(paramB1012(1,1),0,sqrt(VCVB1012(1,1)));
resultsB(i,2)=paramB1012(2,1);
[~,resultsB(i+1,2)]=ztest(paramB1012(2,1),0,sqrt(VCVB1012(2,2)));
resultsB(i,6)=paramB1012(3,1);
[~,resultsB(i+1,6)]=ztest(paramB1012(3,1),0,sqrt(VCVB1012(3,3)));

i=13;

resultsB(i,1)=paramB1112(1,1);
[~,resultsB(i+1,1)]=ztest(paramB1112(1,1),0,sqrt(VCVB1112(1,1)));
resultsB(i,2)=paramB1112(2,1);
[~,resultsB(i+1,2)]=ztest(paramB1112(2,1),0,sqrt(VCVB1112(2,2)));
resultsB(i,4)=paramB1112(3,1);
[~,resultsB(i+1,4)]=ztest(paramB1112(3,1),0,sqrt(VCVB1112(3,3)));
resultsB(i,6)=paramB1112(4,1);
[~,resultsB(i+1,6)]=ztest(paramB1112(4,1),0,sqrt(VCVB1112(4,4)));

i=15;

resultsB(i,1)=paramB1122(1,1);
[~,resultsB(i+1,1)]=ztest(paramB1122(1,1),0,sqrt(VCVB1122(1,1)));
resultsB(i,2)=paramB1122(2,1);
[~,resultsB(i+1,2)]=ztest(paramB1122(2,1),0,sqrt(VCVB1122(2,2)));
resultsB(i,4)=paramB1122(3,1);
[~,resultsB(i+1,4)]=ztest(paramB1122(3,1),0,sqrt(VCVB1122(3,3)));
resultsB(i,6)=paramB1122(4,1);
[~,resultsB(i+1,6)]=ztest(paramB1122(4,1),0,sqrt(VCVB1122(4,4)));
resultsB(i,7)=paramB1122(5,1);
[~,resultsB(i+1,7)]=ztest(paramB1122(5,1),0,sqrt(VCVB1122(5,5)));

i=17;

resultsB(i,1)=paramB2112(1,1);
[~,resultsB(i+1,1)]=ztest(paramB2112(1,1),0,sqrt(VCVB2112(1,1)));
resultsB(i,2)=paramB2112(2,1);
[~,resultsB(i+1,2)]=ztest(paramB2112(2,1),0,sqrt(VCVB2112(2,2)));
resultsB(i,3)=paramB2112(3,1);
[~,resultsB(i+1,3)]=ztest(paramB2112(3,1),0,sqrt(VCVB2112(3,3)));
resultsB(i,4)=paramB2112(4,1);
[~,resultsB(i+1,4)]=ztest(paramB2112(4,1),0,sqrt(VCVB2112(4,4)));
resultsB(i,6)=paramB2112(5,1);
[~,resultsB(i+1,6)]=ztest(paramB2112(5,1),0,sqrt(VCVB2112(5,5)));

i=19;

resultsB(i,1)=paramB1212(1,1);
[~,resultsB(i+1,1)]=ztest(paramB1212(1,1),0,sqrt(VCVB1212(1,1)));
resultsB(i,2)=paramB1212(2,1);
[~,resultsB(i+1,2)]=ztest(paramB1212(2,1),0,sqrt(VCVB1212(2,2)));
```

```
resultsB(i,4)=paramB1212(3,1);
[~,resultsB(i+1,4)]=ztest(paramB1212(3,1),0,sqrt(VCVB1212(3,3)));
resultsB(i,5)=paramB1212(4,1);
[~,resultsB(i+1,5)]=ztest(paramB1212(4,1),0,sqrt(VCVB1212(4,4)));
resultsB(i,6)=paramB1212(5,1);

% Export the tables to an excel file

output_doc='/Users/romaindeharlez/Documents/Volatility models.xlsx';
xlswrite(output_doc,resultsE,'Feuil1','B3:I22');
xlswrite(output_doc,resultsB,'Feuil1','B26:I45');

%% Computing the dynamic conditional correlations parameters

[PARAMETERS,LL,HT,VCV,SCORES,DIAGNOSTICS] = dcc(EU_C,[],1,1,1,1,1,1,1);

Vol_EUQ=zeros(3608,1);
Vol_EUB=zeros(3608,1);
Cov_EU=zeros(3608,1);

for i=1:3608

   Vol_EUQ(i,1)=HT(1,1,i);
   Vol_EUB(i,1)=HT(2,2,i);
   Cov_EU(i,1)=HT(1,2,i);

end

Std_EUQ=sqrt(Vol_EUQ);
Std_EUB=sqrt(Vol_EUB);
EU_corr=ones(3608,1);

for i=1:3608

   EU_corr(i)=Cov_EU(i)/(Std_EUQ(i)*Std_EUB(i));

end

plot(EU_corr)

% Comparing the values with the ones found in the pearson correlation model

pearson_eucorr=zeros(3585,1);
for i=23:3608

   pearson_eucorr(i-22,1)=corr(EU_EQR((i-22):i,1),EU_BR((i-22):i,1));

end
plot(pearson_eucorr)

% Preparation to the chart with both data series

Corr_EU_comparison=[EU_corr(23:3608,1) pearson_eucorr];
figure
plot(Dates(24:3609),Corr_EU_comparison);
dateFormat = 10;
datetick('x',dateFormat)
legend('ADCC corr','Pearson corr')

% And with only one data series
```

```
figure
plot(Dates(24:3609),EU_corr(23:3608,1));
dateFormat = 10;
datetick('x',dateFormat)
legend('ADCC')

% We notice that the general form is quite similar, with lower variation in the DCC model

save('EU_DCC.mat');

%% Testing the fit of the univariate volatility models

% Getting the residuals

EU_Q_res=abs(EU_EQR_C)-Std_EUQ;
EU_B_res=abs(EU_BR_C)-Std_EUB;

% Standardizing the residuals

Std_EUQ_res=EU_Q_res./Std_EUQ;
Std_EUB_res=EU_B_res./Std_EUB;

% Testing for remaining ARCH effect

[hQ4,pQ4] = archtest(Std_EUQ_res);
[hB4,pB4] = archtest(Std_EUB_res);

% Testing for residual autocorrelation

[hQ5,pQ5]=lbqtest(power(Std_EUQ_res,2));
[hB5,pB5]=lbqtest(power(Std_EUB_res,2));
```

2. Multiple linear regression and ADCC forecasts

```
load('Final_EU_DCC.mat');
clearvars -except EU_corr Dates

%% Downloading the needed data

EU_INF=xlsread('/Users/romaindeharlez/Documents/Economics.xlsx','EU INF','B20:B185');
EU_EINF=xlsread('/Users/romaindeharlez/Documents/Economics.xlsx','EU EINF','B12:B67');
EU_GDP=xlsread('/Users/romaindeharlez/Documents/Economics.xlsx','EU GDP','B12:B65');
EU_VOL=xlsread('/Users/romaindeharlez/Documents/Economics.xlsx','EU VOL','B310:B3917');
EU_INT=xlsread('/Users/romaindeharlez/Documents/Economics.xlsx','EU INT','C20:C185');

%% Working on the correlations to get monthly coefficients

day_counter=zeros(166,1);
j=1;
for i=1:3587
    day_counter(j,1)=day_counter(j,1)+1;
    if day(Dates(22+i,1))==15
       j=j+1;
    elseif day(Dates(22+i,1))==14 && weekday(Dates(22+i,1))==6
       j=j+1;
    elseif day(Dates(22+i,1))==13 && weekday(Dates(22+i,1))==6
       j=j+1;
    end
```

```
end

day_accum=length(day_counter);
day_accum(1,1)=day_counter(1,1);

for i=2:length(day_counter)

   day_accum(i,1)=day_accum(i-1,1)+day_counter(i,1);

end

% Applying this formula to every daily DCC coefficient: z = .5 * (log(1 + r) ? log(1 - r))

EU_Z_DCC=zeros(3608,1);

for i=1:3608

   EU_Z_DCC(i)=0.5*(log(1+EU_corr(i,1))-log(1-EU_corr(i,1)));

end

% Computing the average of the fisher transformations of the DCC coeff

EU_Z_average=zeros(3587,1);

for i=1:3587

   EU_Z_average(i,1)=mean(EU_Z_DCC(i:(i+21),1));

end

% Applying the reverse Fisher transformation to get the average DCC coeff
% r = (exp(2 * z) ? 1)/exp(2 * z) + 1)

EU_average_DCC=zeros(3587,1);

for i=1:3587

   EU_average_DCC(i,1)=(exp(2*EU_Z_average(i,1))-1)/(exp(2*EU_Z_average(i,1))+1);

end

% Get the values on the 15th of the month

EU_DCC_monthly=zeros(166,1);

for i=1:166

   EU_DCC_monthly(i,1)=EU_average_DCC(day_accum(i,1),1);

end

% Preparation to the chart

figure
plot(Dates(23:3609),EU_average_DCC);
dateFormat = 10;
datetick('x',dateFormat)
legend('EU ADCC 22d average')
```

```
% chart of expected inflation

expinf_EU_comparison=[EU_DCC_monthly YY_EU_EINF];
figure
plot(expinf_EU_comparison)
legend('EU ADCC','EU EXPINF')


expinf_EU_comparison=[EU_DCC_monthly(2:166,1) diff_EU_EINF];
figure
plot(expinf_EU_comparison)
legend('EU ADCC','diff EU EXPINF')

%% Processing the explanatory variables to get monthly data

% Preparing the vectors for the cubic spline interpolation

X_EU_EINF=zeros(length(EU_EINF),1);
for i=1:length(EU_EINF)
   X_EU_EINF(i)=2+(i-1)*3;
end

X_EU_GDP=zeros(length(EU_GDP),1);
for i=1:length(EU_GDP)
   X_EU_GDP(i)=2+(i-1)*3;
end

Y_EU_EINF=EU_EINF;
Y_EU_GDP=EU_GDP;

XX_EU_EINF=zeros(length(EU_INF),1);
for i=1:length(EU_INF)
   XX_EU_EINF(i,1)=1+i;
end

XX_EU_GDP=zeros(length(EU_INF)-5,1);
for i=1:(length(EU_INF)-5)
   XX_EU_GDP(i,1)=1+i;
end

% Processing cubic spline interpolation

YY_EU_EINF=spline(X_EU_EINF,Y_EU_EINF,XX_EU_EINF);
YY_EU_GDP=spline(X_EU_GDP,Y_EU_GDP,XX_EU_GDP);

% Averaging the implied volatility index

EU_VOL_avg=zeros((length(EU_VOL)-21),1);

for i=22:length(EU_VOL)

   EU_VOL_avg(i-21,1)=mean(EU_VOL((i-21):i,1));

end


EU_VOL_monthly=zeros(166,1);

for i=1:166
```

```
   EU_VOL_monthly(i,1)=EU_VOL_avg(day_accum(i,1),1);

end

%% Preparation of the vectors to the Granger Causality tests

Gger_INF_EU=[EU_DCC_monthly(1:165,1) EU_INF(2:166,1)];
Gger_EINF_EU=[EU_DCC_monthly(1:165,1) YY_EU_EINF(2:166,1)];
Gger_GDP_EU=[EU_DCC_monthly(1:160,1) YY_EU_GDP(2:161,1)];
Gger_INT_EU=[EU_DCC_monthly(1:165,1) EU_INT(2:166,1)];
Gger_VOL_EU=[EU_DCC_monthly(1:165,1) EU_VOL_monthly(2:166,1)];

[hINF,pINF,~,~,~,~] = egcitest(Gger_INF_EU);
[hEINF,pEINF,~,~,~,~] = egcitest(Gger_EINF_EU);
[hGDP,pGDP,~,~,~,~] = egcitest(Gger_GDP_EU);
[hINT,pINT,~,~,~,~] = egcitest(Gger_INT_EU);
[hVOL,pVOL,~,~,~,~] = egcitest(Gger_VOL_EU);

%% Running the regressions

% Checking for collinearity among the explanatory variables

EU_explanatory=[EU_INF(2:161,1) YY_EU_EINF(2:161,1) YY_EU_GDP(2:161,1) EU_INT(2:161,1)
EU_VOL_monthly(2:161,1)];
mdl=fitlm(EU_explanatory,EU_DCC_monthly(1:160,1));
collin=corrcoef(EU_explanatory);
corrplot(EU_explanatory,'testR','on','varNames',{'INF','EXPINF','GDP','INT','VOL'})


% We see that the interest rate variable seems to be highly correlated with
% the expected inflation and other variables

EU_explanatory1=[EU_INF(2:161,1) YY_EU_EINF(2:161,1) YY_EU_GDP(2:161,1)
EU_VOL_monthly(2:161,1)];
collin1=corrcoef(EU_explanatory1);
corrplot(EU_explanatory1,'testR','on')

mdl1=fitlm(EU_explanatory1,EU_DCC_monthly(1:160,1));

% The variable expected inflation still isn't significant, let's try
% without it

EU_explanatory2=[EU_INF(2:161,1) YY_EU_GDP(2:161,1) EU_VOL_monthly(2:161,1)];
collin2=corrcoef(EU_explanatory2);
corrplot(EU_explanatory2,'testR','on')

mdl2=fitlm(EU_explanatory2,EU_DCC_monthly(1:160,1));

% diff of the insignificant variables

diff_EU_INT=diff(EU_INT);
diff_EU_EINF=diff(YY_EU_EINF);
diff_EU_INF=diff(EU_INF);

EU_explanatory3=[EU_INF(2:161,1) YY_EU_GDP(2:161,1) EU_VOL_monthly(2:161,1)
diff_EU_INT(1:160,1) diff_EU_EINF(1:160,1)];
collin3=corrcoef(EU_explanatory3);
corrplot(EU_explanatory3,'testR','on')

mdl3=fitlm(EU_explanatory3,EU_DCC_monthly(1:160,1));
```

```
% Without diff interest rate

EU_explanatory4=[EU_INF(2:161,1) YY_EU_GDP(2:161,1) EU_VOL_monthly(2:161,1)
diff_EU_EINF(1:160,1)];
collin4=corrcoef(EU_explanatory4);
corrplot(EU_explanatory4,'testR','on')

mdl4=fitlm(EU_explanatory4,EU_DCC_monthly(1:160,1));

% Without inflation

EU_explanatory5=[YY_EU_GDP(2:161,1) EU_VOL_monthly(2:161,1) diff_EU_EINF(1:160,1)];
collin5=corrcoef(EU_explanatory5);
corrplot(EU_explanatory5,'testR','on')

mdl5=fitlm(EU_explanatory5,EU_DCC_monthly(1:160,1),'Intercept',false);

% with diff_inflation

EU_explanatory6=[YY_EU_GDP(2:161,1) EU_VOL_monthly(2:161,1) diff_EU_EINF(1:160,1)
diff_EU_INF(1:160,1)];
collin6=corrcoef(EU_explanatory6);
corrplot(EU_explanatory6,'testR','on')

mdl6=fitlm(EU_explanatory6,EU_DCC_monthly(1:160,1));

% without diff_EINF

EU_explanatory7=[YY_EU_GDP(2:161,1) EU_VOL_monthly(2:161,1) diff_EU_INF(1:160,1)];
collin7=corrcoef(EU_explanatory7);
corrplot(EU_explanatory7,'testR','on')

mdl7=fitlm(EU_explanatory7,EU_DCC_monthly(1:160,1));

plotResiduals(mdl5,'probability')
outl = find(mdl5.Residuals.Raw > 0.4);

mdl8 = fitlm(EU_explanatory5,EU_DCC_monthly(1:160,1),'Exclude',outl);

% Tests on the residuals

res=table2array(mdl8.Residuals(1:160,4));
[h1,p1]=lbqtest(res);
[h2,p2]=jbtest(res);
[h3,p3]=archtest(res);

%% Rolling window regressions

rolling_coeff=zeros(250,5);

for i=1:125

   mdl=fitlm(EU_explanatory5(i:(i+35),1:3),EU_DCC_monthly(i:(i+35),1));
   rolling_coeff((2*i-1),1)=table2array(mdl.Coefficients(1,1));
   rolling_coeff((2*i),1)=table2array(mdl.Coefficients(1,4));
   rolling_coeff((2*i-1),2)=table2array(mdl.Coefficients(2,1));
   rolling_coeff((2*i),2)=table2array(mdl.Coefficients(2,4));
   rolling_coeff((2*i-1),3)=table2array(mdl.Coefficients(3,1));
   rolling_coeff((2*i),3)=table2array(mdl.Coefficients(3,4));
```

```
    rolling_coeff((2*i-1),4)=table2array(mdl.Coefficients(4,1));
    rolling_coeff((2*i),4)=table2array(mdl.Coefficients(4,4));
    rolling_coeff((2*i-1),5)=mdl.Rsquared.Adjusted(1,1);

end

xlswrite('/Users/romaindeharlez/Documents/Regression3.xlsx',rolling_coeff,'EU');

%% Forecasting the correlation based on previous results

year=6;

coeff_forecast=zeros((176+(6-year)*24),5);%200+(6-year)*24
dcc_forecast2=zeros((88+(6-year)*12),1);
recap_coeff_6y=zeros(2,4);
j=0;

for i=1:(88+(6-year)*12)

    mdl=fitlm(EU_explanatory5(i:(i+year*12-1),1:3),EU_DCC_monthly((i+1):(i+year*12),1));
    coeff_forecast((2*i-1),1)=table2array(mdl.Coefficients(1,1));
    coeff_forecast((2*i),1)=table2array(mdl.Coefficients(1,4));
    coeff_forecast((2*i-1),2)=table2array(mdl.Coefficients(2,1));
    coeff_forecast((2*i),2)=table2array(mdl.Coefficients(2,4));
    coeff_forecast((2*i-1),3)=table2array(mdl.Coefficients(3,1));
    coeff_forecast((2*i),3)=table2array(mdl.Coefficients(3,4));
    coeff_forecast((2*i-1),4)=table2array(mdl.Coefficients(4,1));
    coeff_forecast((2*i),4)=table2array(mdl.Coefficients(4,4));
    coeff_forecast((2*i-1),5)=mdl.Rsquared.Adjusted(1,1);
    dcc_forecast2(i,1)=mdl.Fitted(((year-1)*12),1);
    j=j+1;
    recap_coeff_6y(1,1)=recap_coeff_6y(1,1)+coeff_forecast((2*i-1),1);
    recap_coeff_6y(2,1)=recap_coeff_6y(2,1)+coeff_forecast((2*i),1);
    recap_coeff_6y(1,2)=recap_coeff_6y(1,2)+coeff_forecast((2*i-1),2);
    recap_coeff_6y(2,2)=recap_coeff_6y(2,2)+coeff_forecast((2*i),2);
    recap_coeff_6y(1,3)=recap_coeff_6y(1,3)+coeff_forecast((2*i-1),3);
    recap_coeff_6y(2,3)=recap_coeff_6y(2,3)+coeff_forecast((2*i),3);
    recap_coeff_6y(1,4)=recap_coeff_6y(1,4)+coeff_forecast((2*i-1),4);
    recap_coeff_6y(2,4)=recap_coeff_6y(2,4)+coeff_forecast((2*i),4);

end

for i=1:2
    for k=1:4
        recap_coeff_6y(i,k)=recap_coeff_6y(i,k)/j;
    end
end



%r_squared((year-1),1)=mean(coeff_forecast(:,5))*2;


xlswrite('/Users/romaindeharlez/Documents/forecasts2.xlsx',coeff_forecast);
xlswrite('/Users/romaindeharlez/Documents/forecast3.xlsx',coeff_forecast);
xlswrite('/Users/romaindeharlez/Documents/forecast4.xlsx',coeff_forecast);
xlswrite('/Users/romaindeharlez/Documents/forecast5.xlsx',coeff_forecast);
xlswrite('/Users/romaindeharlez/Documents/forecast6.xlsx',coeff_forecast);
xlswrite('/Users/romaindeharlez/Documents/EU DCC monthly.xlsx',EU_DCC_monthly);
```

```
% Putting data in the same array to plot

dcc_forecast_plot=zeros(136,6);
dcc_forecast_plot(:,1)=EU_DCC_monthly(25:160,1);
dcc_forecast_plot(:,2)=dcc_forecast2;
dcc_forecast_plot(13:136,3)=dcc_forecast3;
dcc_forecast_plot(25:136,4)=dcc_forecast4;
dcc_forecast_plot(37:136,5)=dcc_forecast5;
dcc_forecast_plot(49:136,6)=dcc_forecast6;

plot(dcc_forecast_plot)

% Preparing the data for the confusion matrixes

forecast_2y=[EU_DCC_monthly(25:160,1) dcc_forecast2];
forecast_3y=[EU_DCC_monthly(37:160,1) dcc_forecast3];
forecast_4y=[EU_DCC_monthly(49:160,1) dcc_forecast4];
forecast_5y=[EU_DCC_monthly(61:160,1) dcc_forecast5];
forecast_6y=[EU_DCC_monthly(73:160,1) dcc_forecast6];

trend_2y=zeros(length(forecast_2y)-1,2);
for i=2:length(forecast_2y)

   if forecast_2y(i,1)>forecast_2y(i-1,1)
      trend_2y(i-1,1)=1;
   else
      trend_2y(i-1,1)=0;
   end

   if forecast_2y(i,2)>forecast_2y(i-1,2)
      trend_2y(i-1,2)=1;
   else
      trend_2y(i-1,2)=0;
   end
end

trend_3y=zeros(length(forecast_3y)-1,2);
for i=2:length(forecast_3y)

   if forecast_3y(i,1)>forecast_3y(i-1,1)
      trend_3y(i-1,1)=1;
   else
      trend_3y(i-1,1)=0;
   end

   if forecast_3y(i,2)>forecast_3y(i-1,2)
      trend_3y(i-1,2)=1;
   else
      trend_3y(i-1,2)=0;
   end
end

trend_4y=zeros(length(forecast_4y)-1,2);
for i=2:length(forecast_4y)

   if forecast_4y(i,1)>forecast_4y(i-1,1)
      trend_4y(i-1,1)=1;
```

```
    else
        trend_4y(i-1,1)=0;
    end

    if forecast_4y(i,2)>forecast_4y(i-1,2)
        trend_4y(i-1,2)=1;
    else
        trend_4y(i-1,2)=0;
    end
end

trend_5y=zeros(length(forecast_5y)-1,2);
for i=2:length(forecast_5y)

    if forecast_5y(i,1)>forecast_5y(i-1,1)
        trend_5y(i-1,1)=1;
    else
        trend_5y(i-1,1)=0;
    end

    if forecast_5y(i,2)>forecast_5y(i-1,2)
        trend_5y(i-1,2)=1;
    else
        trend_5y(i-1,2)=0;
    end
end

trend_6y=zeros(length(forecast_6y)-1,2);
for i=2:length(forecast_6y)

    if forecast_6y(i,1)>forecast_6y(i-1,1)
        trend_6y(i-1,1)=1;
    else
        trend_6y(i-1,1)=0;
    end

    if forecast_6y(i,2)>forecast_6y(i-1,2)
        trend_6y(i-1,2)=1;
    else
        trend_6y(i-1,2)=0;
    end
end

%% creation of the confusion matrix

confusion_2y=zeros(2,2);

for i=1:length(trend_2y)

    if trend_2y(i,1)==0 && trend_2y(i,2)==0
        confusion_2y(2,2)=confusion_2y(2,2)+1;
    elseif trend_2y(i,1)==0 && trend_2y(i,2)==1
        confusion_2y(1,2)=confusion_2y(1,2)+1;
    elseif trend_2y(i,1)==1 && trend_2y(i,2)==0
        confusion_2y(2,1)=confusion_2y(2,1)+1;
    else
        confusion_2y(1,1)=confusion_2y(1,1)+1;
    end
end
```

```
expected_2y=zeros(2,2);
for i=1:2
   for j=1:2

   expected_2y(i,j)=sum(confusion_2y(i,:))*sum(confusion_2y(:,j))/(length(forecast_2y)-1);

   end
end

confusion_3y=zeros(2,2);

for i=1:length(trend_3y)

   if trend_3y(i,1)==0 && trend_3y(i,2)==0
      confusion_3y(2,2)=confusion_3y(2,2)+1;
   elseif trend_3y(i,1)==0 && trend_3y(i,2)==1
      confusion_3y(1,2)=confusion_3y(1,2)+1;
   elseif trend_3y(i,1)==1 && trend_3y(i,2)==0
      confusion_3y(2,1)=confusion_3y(2,1)+1;
   else
      confusion_3y(1,1)=confusion_3y(1,1)+1;
   end
end

expected_3y=zeros(2,2);
for i=1:2
   for j=1:2

   expected_3y(i,j)=sum(confusion_3y(i,:))*sum(confusion_3y(:,j))/(length(forecast_3y)-1);

   end
end

confusion_4y=zeros(2,2);

for i=1:length(trend_4y)

   if trend_4y(i,1)==0 && trend_4y(i,2)==0
      confusion_4y(2,2)=confusion_4y(2,2)+1;
   elseif trend_4y(i,1)==0 && trend_4y(i,2)==1
      confusion_4y(1,2)=confusion_4y(1,2)+1;
   elseif trend_4y(i,1)==1 && trend_4y(i,2)==0
      confusion_4y(2,1)=confusion_4y(2,1)+1;
   else
      confusion_4y(1,1)=confusion_4y(1,1)+1;
   end
end

expected_4y=zeros(2,2);
for i=1:2
   for j=1:2

   expected_4y(i,j)=sum(confusion_4y(i,:))*sum(confusion_4y(:,j))/(length(forecast_4y)-1);

   end
end


confusion_5y=zeros(2,2);
```

```
for i=1:length(trend_5y)

   if trend_5y(i,1)==0 && trend_5y(i,2)==0
      confusion_5y(2,2)=confusion_5y(2,2)+1;
   elseif trend_5y(i,1)==0 && trend_5y(i,2)==1
      confusion_5y(1,2)=confusion_5y(1,2)+1;
   elseif trend_5y(i,1)==1 && trend_5y(i,2)==0
      confusion_5y(2,1)=confusion_5y(2,1)+1;
   else
      confusion_5y(1,1)=confusion_5y(1,1)+1;
   end
end

expected_5y=zeros(2,2);
for i=1:2
   for j=1:2

   expected_5y(i,j)=sum(confusion_5y(i,:))*sum(confusion_5y(:,j))/(length(forecast_5y)-1);

   end
end


confusion_6y=zeros(2,2);

for i=1:length(trend_6y)

   if trend_6y(i,1)==0 && trend_6y(i,2)==0
      confusion_6y(2,2)=confusion_6y(2,2)+1;
   elseif trend_6y(i,1)==0 && trend_6y(i,2)==1
      confusion_6y(1,2)=confusion_6y(1,2)+1;
   elseif trend_6y(i,1)==1 && trend_6y(i,2)==0
      confusion_6y(2,1)=confusion_6y(2,1)+1;
   else
      confusion_6y(1,1)=confusion_6y(1,1)+1;
   end
end

expected_6y=zeros(2,2);
for i=1:2
   for j=1:2

   expected_6y(i,j)=sum(confusion_6y(i,:))*sum(confusion_6y(:,j))/(length(forecast_6y)-1);

   end
end

% computation of the test statistics

test_2y=0;
for i=1:2
   for j=1:2

      test_2y=test_2y+(confusion_2y(i,j)-expected_2y(i,j)).^2/expected_2y(i,j);

   end
end

test_3y=0;
for i=1:2
```

```
   for j=1:2

      test_3y=test_3y+(confusion_3y(i,j)-expected_3y(i,j)).^2/expected_3y(i,j);

   end
end

test_4y=0;
for i=1:2
   for j=1:2

      test_4y=test_4y+(confusion_4y(i,j)-expected_4y(i,j)).^2/expected_4y(i,j);

   end
end

test_5y=0;
for i=1:2
   for j=1:2

      test_5y=test_5y+(confusion_5y(i,j)-expected_5y(i,j)).^2/expected_5y(i,j);

   end
end

test_6y=0;
for i=1:2
   for j=1:2

      test_6y=test_6y+(confusion_6y(i,j)-expected_6y(i,j)).^2/expected_6y(i,j);

   end
end
% Independence hypothesis is rejected in each case, 4 years seems to be the
% best fit

% Computation of the mean absolute percentage error

MAPE_2y=0;
MAPE_3y=0;
MAPE_4y=0;
MAPE_5y=0;
MAPE_6y=0;

for i=1:length(forecast_2y)

   MAPE_2y=MAPE_2y+abs(forecast_2y(i,2)-forecast_2y(i,1))/abs(forecast_2y(i,1));

end
MAPE_2y=MAPE_2y/length(forecast_2y);

for i=1:length(forecast_3y)

   MAPE_3y=MAPE_3y+abs(forecast_3y(i,2)-forecast_3y(i,1))/abs(forecast_3y(i,1));

end
MAPE_3y=MAPE_3y/length(forecast_3y);

for i=1:length(forecast_4y)
```

```
   MAPE_4y=MAPE_4y+abs(forecast_4y(i,2)-forecast_4y(i,1))/abs(forecast_4y(i,1));

end
MAPE_4y=MAPE_4y/length(forecast_4y);

for i=1:length(forecast_5y)

   MAPE_5y=MAPE_5y+abs(forecast_5y(i,2)-forecast_5y(i,1))/abs(forecast_5y(i,1));

end
MAPE_5y=MAPE_5y/length(forecast_5y);

for i=1:length(forecast_6y)

   MAPE_6y=MAPE_6y+abs(forecast_6y(i,2)-forecast_6y(i,1))/abs(forecast_6y(i,1));

end
MAPE_6y=MAPE_6y/length(forecast_6y);

plot(forecast_6y)
```

3. ARIMAX model for ADCC forecasts

```
load('EU_reg.mat')

[h,pValue] = lmctest(EU_DCC_monthly);
dcc_forecast_biased=zeros(88,1);
dcc_forecast_unbiased=zeros(88,1);
Model=arima(1,0,0);
mean_bia=0;
coeff=zeros(1,3);

for i=1:88


mdl=estimate(Model,EU_DCC_monthly((i+1):(i+72),1),'X',EU_explanatory5(i:(i+71),1),'print',false,'Y0',
EU_DCC_monthly(i,1));

new_forecasts=infer(mdl,EU_DCC_monthly((i+1):(i+72),1),'Y0',EU_DCC_monthly(i,1),'X',EU_explana
tory5(i:(i+71),1));
  dcc_forecast_biased(i,1)=new_forecasts(72,1);
  coeff(1,1)=coeff(1,1)+table2array(mdl.AR);
  coeff(1,2)=coeff(1,2)+(mdl.Constant);
  coeff(1,3)=coeff(1,3)+(mdl.Beta);

  if i>1

     mean_bia=((i-1)/i)*mean_bia+(1/i)*(dcc_forecast_biased(i-1,1)-EU_DCC_monthly(i+71,1));

  end

  dcc_forecast_unbiased(i,1)=dcc_forecast_biased(i,1)-mean_bia;

end

coeff_avg=coeff/88;

forecast_6y=[EU_DCC_monthly(73:160,1) dcc_forecast_unbiased];
figure
```

```
plot(forecast_6y);
legend('Actual EU ADCC','Forecasts')

% Computation of the MAPE

MAPE_arimax=0;

for i=1:length(dcc_forecast_unbiased)

   MAPE_arimax=MAPE_arimax+abs(dcc_forecast_unbiased(i,1)-
EU_DCC_monthly(i+72,1))/abs(EU_DCC_monthly(i+72,1));

end
MAPE_arimax=MAPE_arimax/length(dcc_forecast_unbiased);

% Preparing data for confusion matrixes

forecast_6y=[EU_DCC_monthly(73:160,1) dcc_forecast_unbiased];

trend_6y=zeros(length(forecast_6y)-1,2);
for i=2:length(forecast_6y)

   if forecast_6y(i,1)>forecast_6y(i-1,1)
      trend_6y(i-1,1)=1;
   else
      trend_6y(i-1,1)=0;
   end

   if forecast_6y(i,2)>forecast_6y(i-1,2)
      trend_6y(i-1,2)=1;
   else
      trend_6y(i-1,2)=0;
   end
end

confusion_6y=zeros(2,2);

for i=1:length(trend_6y)

   if trend_6y(i,1)==0 && trend_6y(i,2)==0
      confusion_6y(2,2)=confusion_6y(2,2)+1;
   elseif trend_6y(i,1)==0 && trend_6y(i,2)==1
      confusion_6y(1,2)=confusion_6y(1,2)+1;
   elseif trend_6y(i,1)==1 && trend_6y(i,2)==0
      confusion_6y(2,1)=confusion_6y(2,1)+1;
   else
      confusion_6y(1,1)=confusion_6y(1,1)+1;
   end
end

expected_6y=zeros(2,2);
for i=1:2
   for j=1:2

   expected_6y(i,j)=sum(confusion_6y(i,:))*sum(confusion_6y(:,j))/(length(forecast_6y)-1);

   end
end

test_6y=0;
```

```
for i=1:2
   for j=1:2

      test_6y=test_6y+(confusion_6y(i,j)-expected_6y(i,j)).^2/expected_6y(i,j);

   end
end

save('test_arimax_EU.mat')
```

4. Forecasts of the assets volatilities

```
load('EU_reg.mat');
clearvars -except day_counter day_accum
load('Final_EU_DCC.mat');
clearvars -except EU_EQR_C EU_BR_C Vol_EUB Vol_EUQ day_counter day_accum

%% Forecasts of the volatility
% EU Equities

length_forecast=sum(day_counter(73:160,1));

EUQ_forecast=zeros(length_forecast,1);
model=gjr('GARCH',NaN,'ARCH',NaN,'Leverage',NaN,'Distribution','t');
j=1;

obs_window=EU_EQR_C(22:(day_accum(72,1)+21),1);
mdl=estimate(model,obs_window);
Var_rolling=infer(mdl,obs_window);
[Vsim,~] = simulate(mdl,day_counter(73,1),'NumPaths',500,'E0',obs_window,'V0',Var_rolling);
EUQ_forecast(j:(j+day_counter(73,1)-1),1)=mean(Vsim,2);
j=j+day_counter(73,1);

for i=2:88

   obs_window=EU_EQR_C((day_accum(i-1,1)+22):(day_accum(i+71,1)+21),1);
   mdl=estimate(model,obs_window);
   Var_rolling=infer(mdl,obs_window);
   [Vsim,~] = simulate(mdl,day_counter(i+72,1),'NumPaths',500,'E0',obs_window,'V0',Var_rolling);
   EUQ_forecast(j:(j+day_counter(i+72,1)-1),1)=mean(Vsim,2);
   j=j+day_counter(i+72,1);

end

EQ_avg=zeros(1892,1);

for i=1:1892

   EQ_avg(i,1)=mean(EUQ_forecast(i:(i+21),1));

end

EQ_forecast_monthly=zeros(88,1);

for i=73:160

   EQ_forecast_monthly((i-72),1)=EQ_avg((day_accum(i,1)-1574),1);
```

```
end

% EU Bonds

EUB_forecast=zeros(length_forecast,1);
model=gjr('GARCH',NaN,'ARCH',NaN,'Leverage',NaN,'Distribution','t');
j=1;

obs_window=EU_BR_C(22:(day_accum(72,1)+21),1);
mdl=estimate(model,obs_window);
Var_rolling=infer(mdl,obs_window);
[Vsim,~] = simulate(mdl,day_counter(73,1),'NumPaths',500,'E0',obs_window,'V0',Var_rolling);
EUB_forecast(j:(j+day_counter(73,1)-1),1)=mean(Vsim,2);
j=j+day_counter(73,1);

for i=2:88

   obs_window=EU_BR_C((day_accum(i-1,1)+22):(day_accum(i+71,1)+21),1);
   mdl=estimate(model,obs_window);
   Var_rolling=infer(mdl,obs_window);
   [Vsim,~] = simulate(mdl,day_counter(i+72,1),'NumPaths',500,'E0',obs_window,'V0',Var_rolling);
   EUB_forecast(j:(j+day_counter(i+72,1)-1),1)=mean(Vsim,2);
   j=j+day_counter(i+72,1);

end

plot(EUB_forecast)
figure(2);
plot(Vol_EUB)


B_avg=zeros(1892,1);

for i=1:1892

   B_avg(i,1)=mean(EUB_forecast(i:(i+21),1));

end

figure(3);
plot(B_avg)


B_forecast_monthly=zeros(88,1);

for i=73:160

   B_forecast_monthly((i-72),1)=B_avg((day_accum(i,1)-1574),1);

end


%% Exporting the data to an excel file

filename='/Users/romaindeharlez/Documents/VaR optimization.xlsx';

% Averaging the assets volatilities

Vol_EUQ_avg=zeros(3587,1);
Vol_EUB_avg=zeros(3587,1);
```

```
for i=1:3587

   Vol_EUQ_avg(i,1)=mean(Vol_EUQ(i:(i+21),1));
   Vol_EUB_avg(i,1)=mean(Vol_EUB(i:(i+21),1));

end

Vol_EUQ_monthly=zeros(166,1);
Vol_EUB_monthly=zeros(166,1);

for i=1:166

   Vol_EUQ_monthly(i,1)=Vol_EUQ_avg(day_accum(i,1),1);
   Vol_EUB_monthly(i,1)=Vol_EUB_avg(day_accum(i,1),1);

end

EU_forecasts_global=[EQ_forecast_monthly B_forecast_monthly dcc_forecast6
Vol_EUQ_monthly(73:160) Vol_EUB_monthly(73:160) EU_DCC_monthly(73:160)];

xlswrite(filename,EU_forecasts_global);

Vol_EUQ_comparison=[Vol_EUQ_monthly(73:160) EQ_forecast_monthly];
figure
plot(Vol_EUQ_comparison);
legend('EU Equities Volatility','Forecasts')

Vol_EUB_comparison=[Vol_EUB_monthly(73:160) B_forecast_monthly];
figure
plot(Vol_EUB_comparison);
legend('EU Bonds Volatility','Forecasts')

% Computing MAPE

MAPE_Vol_EQforecast=0;
MAPE_Vol_Bforecast=0;

for i=1:length(EQ_forecast_monthly)

   MAPE_Vol_EQforecast=MAPE_Vol_EQforecast+abs(EQ_forecast_monthly(i,1)-
Vol_EUQ_monthly(i+72,1))/abs(Vol_EUQ_monthly(i+72,1));
   MAPE_Vol_Bforecast=MAPE_Vol_Bforecast+abs(B_forecast_monthly(i,1)-
Vol_EUB_monthly(i+72,1))/abs(Vol_EUB_monthly(i+72,1));

end

MAPE_Vol_EQforecast=MAPE_Vol_EQforecast/length(EQ_forecast_monthly);
MAPE_Vol_Bforecast=MAPE_Vol_Bforecast/length(EQ_forecast_monthly);
```

5. Optimization function

```
%% Optimization function

function f = myObjective(xin,a,b,c)
f = xin(1)^2*a+xin(2)^2*b+xin(1)*xin(2)*c*sqrt(a)*sqrt(b);
```

6. VaR optimization

```
%% Computation of the equally weighted portfolio VaR

load('test_arimax_EU.mat');
clearvars -except dcc_forecast_unbiased
load('EU_Vol_forecast.mat');
EQW_port_VaR=zeros(88,1);

EU_forecasts_global(:,3)=dcc_forecast_unbiased;

for i=1:88


EQW_port_VaR(i,1)=sqrt((0.5).^2*EU_forecasts_global(i,4)+(0.5).^2*EU_forecasts_global(i,5)+0.25*E
U_forecasts_global(i,6)*sqrt(EU_forecasts_global(i,4))*sqrt(EU_forecasts_global(i,5)))*1.662;

end

weights=zeros(88,2);
Aeq=ones(1,2);
Beq=1;
fval=zeros(88,1);

for i=1:88

a=EU_forecasts_global(i,1);
b=EU_forecasts_global(i,2);
c=EU_forecasts_global(i,3);
xin=[1;1];

f = @(xin)myObjective(xin,a,b,c);
[weights(i,1:2),fval(i,1)] = fmincon(f,xin,[],[],Aeq,Beq,[0.4,0.4],[0.6,0.6]);

end

VaR_optim=zeros(88,1);
for i=1:88


VaR_optim(i,1)=sqrt((weights(i,1)).^2*EU_forecasts_global(i,4)+(weights(i,2)).^2*EU_forecasts_global
(i,5)+weights(i,1)*weights(i,2)*EU_forecasts_global(i,6)*sqrt(EU_forecasts_global(i,4))*sqrt(EU_foreca
sts_global(i,5)))*1.662;

end

variance_optim=zeros(88,1);
variance_eqw=zeros(88,1);
```

```
for i=1:88

   variance_optim(i,1)=(VaR_optim(i,1)/1.662)^2;
   variance_eqw(i,1)=(EQW_port_VaR(i,1)/1.662)^2;

end

meanVaR_eqw=sqrt(mean(variance_eqw))*1.662;
meanVaR_optim=sqrt(mean(variance_optim))*1.662;

% Computing the paired t-test

diff_VaR=EQW_port_VaR-VaR_optim;
squared_diff=diff_VaR.*diff_VaR;
squared_sum_diff=sum(squared_diff);
sum_squared_diff=sum(diff_VaR)^2;
mean_diff=mean(diff_VaR);
variance_diff=(squared_sum_diff-sum_squared_diff/88)/87;
paired_ttest=mean_diff/(sqrt(variance_diff)/sqrt(88));

% Saving

%save('VaR_optim.mat');

%% Computation of the Conditional VaR

% Computing the monthly returns

clearvars -except weights meanVaR_eqw meanVaR_optim
load('EU_reg.mat');
clearvars -except day_counter day_accum weights meanVaR_eqw meanVaR_optim
load('Final_EU_DCC.mat');
clearvars -except EU_EQR_C EU_BR_C Vol_EUB Vol_EUQ day_counter day_accum EU_B EU_EQ
weights meanVaR_eqw meanVaR_optim

monthly_EUB=zeros(89,1);
monthly_EUQ=zeros(89,1);
monthly_EUQ(1,1)=EU_EQ((day_accum(73,1)+22),1);
monthly_EUB(1,1)=EU_B((day_accum(73,1)+22),1);

for i=2:89

   monthly_EUQ(i,1)=EU_EQ((day_accum((i+72),1)+22),1);
   monthly_EUB(i,1)=EU_B((day_accum((i+72),1)+22),1);

end

monthly_EUQR=zeros(88,1);
monthly_EUBR=zeros(88,1);

for i=1:88

   monthly_EUQR(i,1)=log(monthly_EUQ(i+1,1)/monthly_EUQ(i,1));
   monthly_EUBR(i,1)=log(monthly_EUB(i+1,1)/monthly_EUB(i,1));

end

% Computing the portfolio returns of the equally weighted and VaR optimized weights
```

```
EQW_returns=zeros(88,1);
VaRoptim_returns=zeros(88,1);

for i=1:88

   EQW_returns(i,1)=0.5*monthly_EUQR(i,1)+0.5*monthly_EUBR(i,1);
   VaRoptim_returns(i,1)=weights(i,1)*monthly_EUQR(i,1)+weights(i,2)*monthly_EUBR(i,1);

end

% Finding the distribution of the losses
losses=zeros(88,1);
j=1;

for i=1:88

   if EQW_returns(i,1)<=0

      losses(j,1)=EQW_returns(i,1);
      j=j+1;

   end

end

final_losses=losses(1:29,1);
% Testing the normality of the distribution

[hEQW,pEQW]=jbtest(EQW_returns);
[hVaR,pVaR]=jbtest(VaRoptim_returns);

% Conditional VaR

CondVaR_eqw=0;
n_eqw=0;
CondVaR_optim=0;
n_optim=0;
paired_trial=zeros(88,2);

for i=1:88

   if EQW_returns(i,1)<(-meanVaR_eqw)

      CondVaR_eqw=CondVaR_eqw+EQW_returns(i,1);
      n_eqw=n_eqw+1;
      paired_trial(i,1)=EQW_returns(i,1)+meanVaR_eqw;

   end

   if VaRoptim_returns(i,1)<(-meanVaR_optim)

      CondVaR_optim=CondVaR_optim+VaRoptim_returns(i,1);
      n_optim=n_optim+1;
      paired_trial(i,2)=VaRoptim_returns(i,1)+meanVaR_optim;

   end

end

CondVaR_optim=CondVaR_optim/n_optim;
```

```
CondVaR_eqw=CondVaR_eqw/n_eqw;

% Paired t-test on the conditional VaR

diff_CVaR=paired_trial(1:88,1)-paired_trial(1:88,2);
squared_diff_CVaR=diff_CVaR.*diff_CVaR;
squared_sum_diffCVaR=sum(squared_diff_CVaR);
sum_squared_diffCVaR=sum(diff_CVaR)^2;
mean_diff_CVaR=mean(diff_CVaR);
variance_diff_CVaR=(squared_sum_diffCVaR-sum_squared_diffCVaR/88)/87;
paired_ttest_CVaR=mean_diff_CVaR/(sqrt(variance_diff_CVaR)/sqrt(88));


%% Saving the results

%save('Optim_Condvar.mat');
```