TASK

It is well-known that the dynamics of many financial time series exhibit discontinuities. Relationships between the key factors which are true today may not be true tomorrow. The moments when the key relationships change are called the times of “structural change”. Apply the Hidden Markov Model methodology to identify the structural change moments for the USD/EUR exchange rate. The analysis must be exploratory and robust. As such, it should not employ very complicated models relying on too many assumptions.

SOLUTION

APPLICATION OF A HIDDEN MARKOV MODEL TO LOG-RETURNS OF THE USD/EUR EXCHANGE RATE: AN ORIGINAL IMPLEMENTATION IN MATLAB

INTRODUCTION

This study analyzes a particular financial time series within the framework of Hidden Markov Models (HMM). An HMM is a model of a stochastic process which states the following: the distribution of the process tomorrow given its value today depends on the current state of the world, which is not observed. There are at least two distinct states of the world. The world jumps from one state to another according to a continuous-time or discrete-time Markov chain. The conditional distribution of the process may be a simple function of the state of the world. It may be a Normal or Poisson distribution. However, the hidden nature of the state makes the model complicated. The model can be estimated using an iterative version of the maximum likelihood method (like EM algorithm) or Bayesian methods.

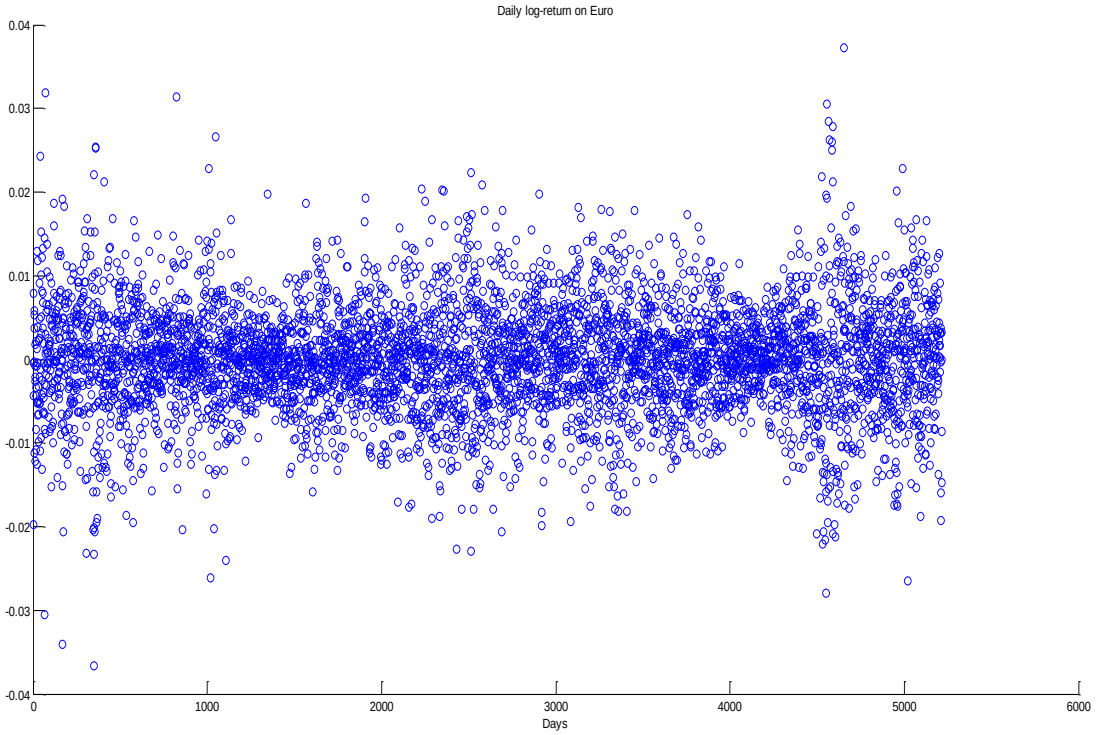
For our study, we choose daily log-returns of the USD/EUR exchange rate observed during the period May 17, 1991 – May 16, 2011. We prefer to focus on log-returns instead of prices because they have a much better chance of being stationary processes, amenable to HMM modeling. Focusing on daily observations allows us to gather enough data to estimate complex models, with more than a dozen of parameters sometimes. The USD/EUR exchange rate is defined as the amount of dollars per one euro.

We feel the need to develop a reasonably complex model for USD/EUR for **risk management purposes**. Most likely, the distribution of USD/EUR changes over time. Understanding its current expectation and current volatility allows to hedge short-term derivatives on USD/EUR more efficiently. By short-term derivatives we mean those with tenors up to 6 months. Such derivatives are very sensitive to fluctuations in the volatility, typically... Sticking to traditional autoregressive models is likely to make hedging less accurate. Oftentimes the distribution of the exchange rate changes abruptly because of the news in the market. A traditional model would not pick up a change quickly enough, which would lead to miscalculating the probabilities of various scenarios of the profit & loss on the derivative. For example an AR-GARCH model would quickly pick an upward jump in the volatility of the exchange rate but it would fail to capture a downward jump in the volatility quickly enough.

Risk management is an important part of market making and proprietary trading. To the best of our knowledge, more and more hedge funds are currently experimenting with regime switching models for that purpose.

Now let us get back to the modeling issues. We know that USD/EUR must exhibit slightly different types of behavior under different global conditions. So the global conditions could be the hidden state in our model. However, the exact nature of variation of log-returns with the state is not clear. And we need to figure it out

before we build the model. To develop intuition, we settle for the obvious. We plot the time series of the log-returns, hoping to see something. The plot is placed below (all the plots in this study are also attached in good quality).



Log-returns appear to have a relatively stable conditional mean. This characteristic may be invariant with the state. On the other hand, the volatility of log-returns exhibits random behavior, jumping to relatively high levels during certain historic periods and dropping back on some other days. The volatility is to become the only characteristic changing from one state to another in our model.

We define the model in the following way. Let R_t be the log-return at time t. There are K different states of the world, numbered 1, 2, ..., K. The state on day 1 is distributed according to probabilities Pi = (pi_1, pi_2, ..., pi_K). These probabilities are called the initial distribution. The state of the world follows a discrete-time Markov chain with the transition matrix A = (a_ij), where all a_ij can be positive, generally speaking. If at time t the Markov chain is in state k, then

$$R_t = Beta_0 + Beta_1 * R_{t-1} + ... + Beta_p * R_{t-p} + sigma_k * eps_t,$$

where eps_t is a standard normal variable, which is independent of everything else. It is important that the conditional volatility strictly increases with the state:

$$sigma_1 < sigma_2 < ... < sigma_K.$$

The conditional volatility is the only thing that distinguishes the states (of the world).

We consider 9 different specifications of the model, corresponding to different combinations of the number of lags p and the number of states K:

model 1: p = 0, K = 2,
model 2: p = 1, K = 2,
model 3: p = 2, K = 2,

model 4: p = 0, K = 3,
model 5: p = 1, K = 3,
model 6: p = 2, K = 3,
model 7: p = 0, K = 4,
model 8: p = 1, K = 4,
model 9: p = 2, K = 4.

Model 9 is most flexible but requiring the most data to estimate. Model 1 is most parsimonious. We will have to use model selection tools to identify the optimal performer among the 9 candidates. One of the most popular model selection tools are cross-validation and Akaike information criterion (AIC). We will base our decisions on cross-validation but will keep an eye on the AIC statistics as a sanity check.

CROSS-VALIDATION & ESTIMATION

The period of May 17, 1991 – May 16, 2011 presents 5217 data points of daily log-returns on euro. We split those data points into two periods: the estimation-validation period and the testing period. Due to the slowness of the estimation procedure, the estimation-validation period is chosen to be days 3 through 2002, which amounts to the total of 2000 data points. We need the first two days to get the values of lags R_{t-1} and R_{t-2} in models 2, 3, 5, 6, 8 and 9. Therefore, they do not count towards the training-validation period. The remaining data, corresponding to days 2003 through 5217, is the testing period. These data are used to study the predictive performance of the models estimated earlier. Both estimation-validation and testing periods are big enough to make the results of estimation, optimal model identification and predictive diagnostics accurate.

Implicit in our analysis is the assumption that the dynamics of the exchange rate is the same on any subinterval of the estimation-validation and testing periods. Loosely speaking, if a model is adequate for one relatively large subinterval, it is adequate for any other relatively large subinterval.

Our analysis has been done with the help of Matlab functions `HMM_Analysis`(), `HMM_Estimation`(), `HMM_Estimation_OneIniGuess`(), `HMM_VolatilityPrediction`() and `HMM_Viterbi`(), build specifically for this study. The functions contain numerous comments, explaining every step and mathematical manipulation. The scripts are attached. Also, their body is placed in appendix B. The key Matlab commands required to leverage the scripts are placed in appendix A, whereas the complete Matlab session with all the output is placed in appendix C.

The first stage of our analysis is 2-fold cross-validation. We split the estimation-validation set into two equally-sized blocks. Block 2 follows block 1 chronologically. We estimate each of the 9 models on block 1. As an estimation method we use the **EM-algorithm**. We extend the version of the EM-algorithm described in chapter "Sequential Data" of Bishop's "Pattern Recognition and Machine Learning" for our case. Our model is slightly more general than the framework treated by Bishop because of the auto-regressive coefficients Beta_1, ... , Beta_p.

The EM-algorithm is an efficient tool. However, it has the flaw of converging to a wrong solution if the starting values of the parameters are close to a local maximum of the likelihood function. Therefore, at each estimation step, we run the EM-algorithm with **4 different sets of starting values**. How those starting values are chosen can be seen on lines 27 – 44 of script `HMM_Estimation_OneIniGuess`(). We then choose the parameter estimates that maximize the likelihood function, ignoring the other 3 sets of the parameter estimates. We would have preferred to use 10 different starting values instead of 4. However, that would

have made the whole analysis quite slow... The EM estimation over different starting values is implemented in scripts `HMM_Estimation`() and `HMM_Estimation_OneIniGuess`().

Recall that each model is estimated on data bock 1. If model i is correct and its parameter estimates are accurate, then model i must predict volatility in block 2 relatively accurately. We measure the discrepancy between the true and predicted volatilities in the following fashion. For each day in block 2, we predict the volatility using the data from the previous days in block 2 and the parameter estimates from block 1. We use the **forward algorithm**, described in chapter "Sequential Data" of Bishop's "Pattern Recognition and Machine Learning" and section 2.1 of Fraser's "Hidden Markov Models and Dynamical Systems". Suppose that sigma_hat_t is the predicted volatility on day t and Beta_hat_0, ... , Beta_hat_p are the parameter estimates. Then we standardize the log return with the following transformation:

*R_stand_t = (R_t - Beta_hat_0 + Beta_ hat_1 * R_{t-1} + ... + Beta_ hat_p * R_{t-p}) / sigma_hat_t.*

Essentially, we are building the estimate of residual eps_t.

If our model is correct, than the variance of the standardized residuals on block 2 must be approximately 1. However, the model may not be that great. On average, it may predict the variance to be M times bigger than the truth or M times smaller than the truth. In both those cases, we want the measure of prediction error to be the same, for symmetry reasons. Therefore we define the prediction error as

PE = abs(log(VAR[R_stand_t])).

We see that if the prediction of variance is M times bigger than the truth, on average, then PE = log(M). And we see that if the prediction of variance is M times smaller than the truth, on average, then PE = abs(-log(M)) = log(M). The smaller PE is the better the model is. We calculate PE for each of the 9 models, recording the numbers as important information about the goodness of the models.

All the calculations above correspond to only a half of the cross-validation procedure. Now we perform their symmetric counterpart. We estimate each of the 9 candidate models on block 2 and calculate their PE on block 1. Now we have two PE numbers for each candidate model. We average those numbers. This is the final cross-validation estimate of the prediction error for each model. That model is considered best which has the smallest cross-validation error. Below are displayed the cross-validation errors, the aggregate ones and the errors for each block.

CV_Errors =

```
0.3038
0.3037
0.3023
0.1905
0.1922
0.1914
0.3213
0.3354
0.3211
```

CV_Errors_For_Blocks =

```
0.5950   0.0126
0.5953   0.0120
0.5925   0.0122
0.3612   0.0199
0.3611   0.0233
0.3596   0.0233
0.6112   0.0313
0.6141   0.0567
0.6090   0.0331
```

We see that **model 4 performs better than the others**, according to cross-validation. Lets us remind you that the model equation is:

$$R_t = Beta_0 + sigma_k * eps_t,$$

where sigma_k takes three values: sigma_1 < sigma_2 < sigma_3. Here it seems that 3 states of the world perform much better than 2 or 4 states of the world. We will see shortly if the cross-validation choice of the best model is consistent with the AIC choice of the best model.

As the second stage of our analysis, we estimate each of the 9 candidate models on the whole estimation-validation data set. Of course, we are most keen to learn the parameter estimates of the optimal model (model 4). But we also need the estimates of other models to get the AIC scores as by-products of estimation and to perform various other diagnostics. The estimation is done using the EM-algorithm. The estimates of model 4 are displayed below.

$$Beta\_0 = -1.1700e\text{-}005,$$

$$sigma\_1 = 0.0037,$$
$$sigma\_2 = 0.0057,$$
$$sigma\_3 = 0.0126,$$

$$Pi = (0, 1, 0),$$

$$A = \begin{matrix} 0.9864 & 0.0059 & 0.0077 \\ 0.0031 & 0.9735 & 0.0234 \\ 0.0180 & 0.1774 & 0.8046 \end{matrix}$$

The estimate of Beta_0 is insignificant, as manifested by its standard error of 1.3943e-004. This leads us to look more carefully into why more flexible models 5 and 6 have not performed as well as model 4. As we know, the only difference between model 4 and the other two models is the presence of autoregressive coefficients. We discover that **autoregressive coefficients are statistically insignificant** for models 5 and 6. In particular, this is how the autoregressive estimates and their standard errors look for model 6:

```
Model_6.AutoRegressCoeff =

   -0.0000
   -0.0262
    0.0193

Model_6.AutoRegressCoeff_StdErrors =

    0.0001
    0.0224
    0.0223
```

We conclude that the true autoregressive coefficients must be 0, which is **consistent with financial theory**. The complete estimation results can be found in appendix C. As the last touch, we generate AIC scores for the 9 candidate models and see what they suggest.

```
AkaikeInfoCriteria =

   -7.4382
   -7.4389
   -7.4387
   -7.4655
   -7.4667
   -7.4665
   -7.4769
   -7.4782
   -7.4776
```

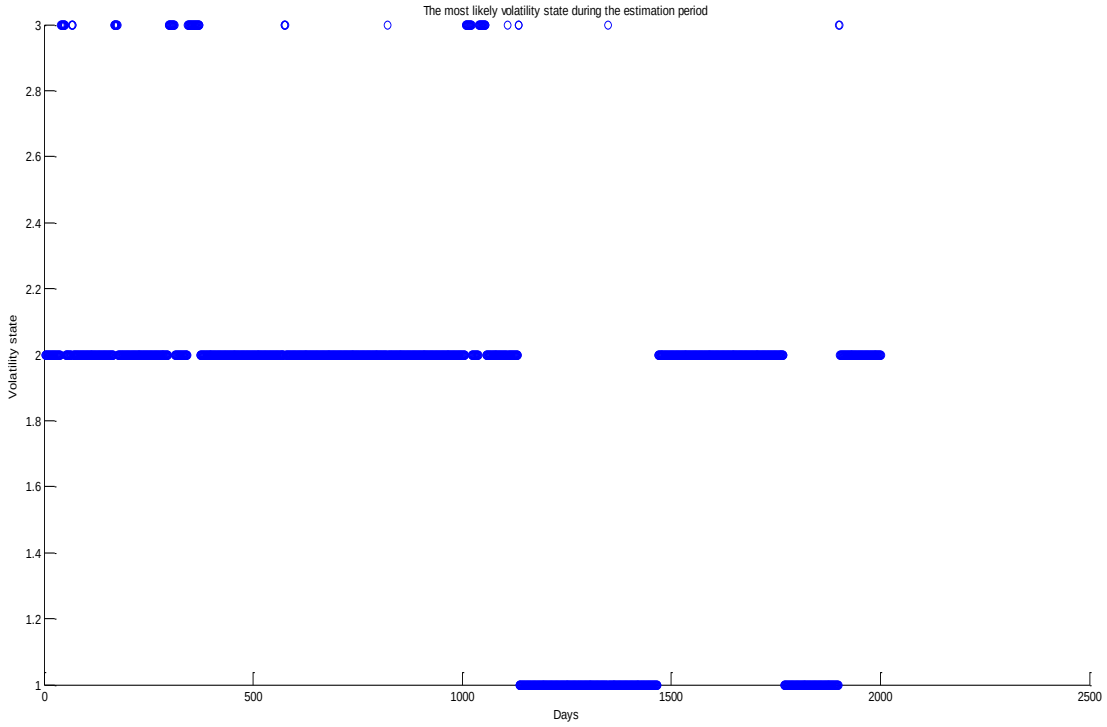
The conclusion of the Akaike criterion is different from that of cross-validation. The Akaike criterion seems to favor model 8 (p = 1, K = 4). However, the Akaike scores are so close to one another that the criterion seems to be less accurate than cross-validation in our case.

THE MAXIMUM A PRIORI STATE SEQUENCE AND VITERBY ALGORITHM

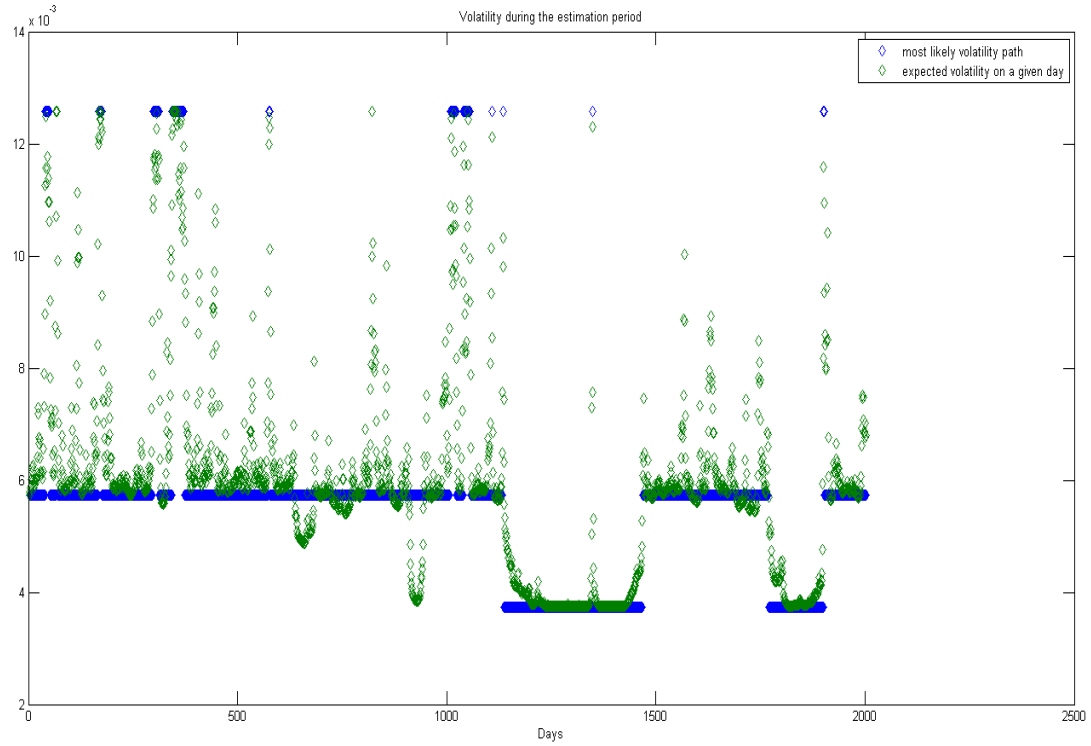
Ok. Now that we have found the model to love, we can exploit it fully to get insights into the states of the world and the associated volatility dynamics during the estimation-validation period. We do that with the help of the Viterbi algorithm. It is described well in section 2.3 of Fraser's book. The description in Bishop's book is somewhat cryptic. We implemented the Viterbi algorithm in script `HMM_Viterbi`(), containing numerous comments.

The graph below displays the **maximum a posteriori** (MAP) sequence of states. In plain English, this is the most likely scenario of the states of the world given the data. We note, that if S_t is the component of MAP corresponding to day t, this does not mean that S_t is the most likely state on day t, given the data. The *sequence* is most likely but any of its components is not necessarily most likely. This is a consequence of the fact that, even if states S_{t-1} and S_t are most likely given the data, the transition from S_{t-1} to S_t may not be likely according to the estimate of transition matrix A.

Identification of the maximum a posteriori state sequence and the corresponding volatilities is the third stage of our analysis.



We see that the world starts at state 2 and spends quite a bit of time there. From time to time there are moves to the other states, where perceptible amount of time is spent. The world spends the least amount of time at state 3 (the high-volatility environment). Also we note that there have been moves between each pair of states. As the next step, we want to see what this implies for the volatility. On the same graph we display 1) the volatility at the current state of the world according to MAP and 2) the expected volatility on each separate day given the data. The latter is calculated using conditional probabilities which have been generated as a by-product of EM estimation. For details see script `HMM_Estimation_OneIniGuess`().



As can be seen above, the expected volatility line reflects the most likely state sequence most of the time, but not always. **This is an illustration of the property stating that any of the components of the most likely state sequence is not necessarily most likely on that day**.

PREDICTIVE DIAGNOSTICS

As the fourth stage of our analysis, we test predictive performance of all 9 models on a fresh data set, which corresponds to the testing period. We want to see if exploiting cross-validation has led to the right choice of the model. We also want to see how close the predictive performance of runners-up is. The predictive performance is measured using the good old PE measure as well the percentage of 1-sigma, 2-sigma and 3-sigma events that have happened in the testing period according to the predictions of the models. The results are displayed below.

PredictionInterval_AbsLogError (PE) =

0.1956
0.1983
0.1950
0.2008
0.2056
0.2003
0.2142
0.2195
0.2182

PredictionInterval_VarOfStandResiduals =

0.8224
0.8201
0.8228
0.8180
0.8142
0.8185
0.8072
0.8029

0.8040

PredictionInterval_Perc_Of_1STD_Events =

0.3066
0.3059
0.3049
0.3076
0.3069
0.3052
0.2876
0.2886
0.2883

PredictionInterval_Perc_Of_2STD_Events =

0.0093
0.0090
0.0093
0.0028
0.0028
0.0028
0.0121
0.0090
0.0104

PredictionInterval_Perc_Of_3STD_Events =

1.0e-003 *

0.6913
0.6913
0.6913
0
0
0
0
0
0

Let's summarize the above. **In terms of PE, model 4 is not the best one on the testing set but it is close to the best performers**. **Model 4 updates the volatility forecast in such a way that 1-sigma events happen as they should, approximately 32% of the time**. We do note that a big subset of the candidate models performs quite well here: models 1 trough 6. On the other hand, all the models are disgraceful when it comes to counting 2-sigma events. Here model 4 is much worse than models 1 - 3 and 7 - 9, but those are not great either. **The only models that lead to near-correct percentages of 3-sigma events are models 1 – 3 (two states of the world)**. Note that the best model according to the Akaike information criterion (model 8) performs worse than the best model according to cross-validation (model 4).

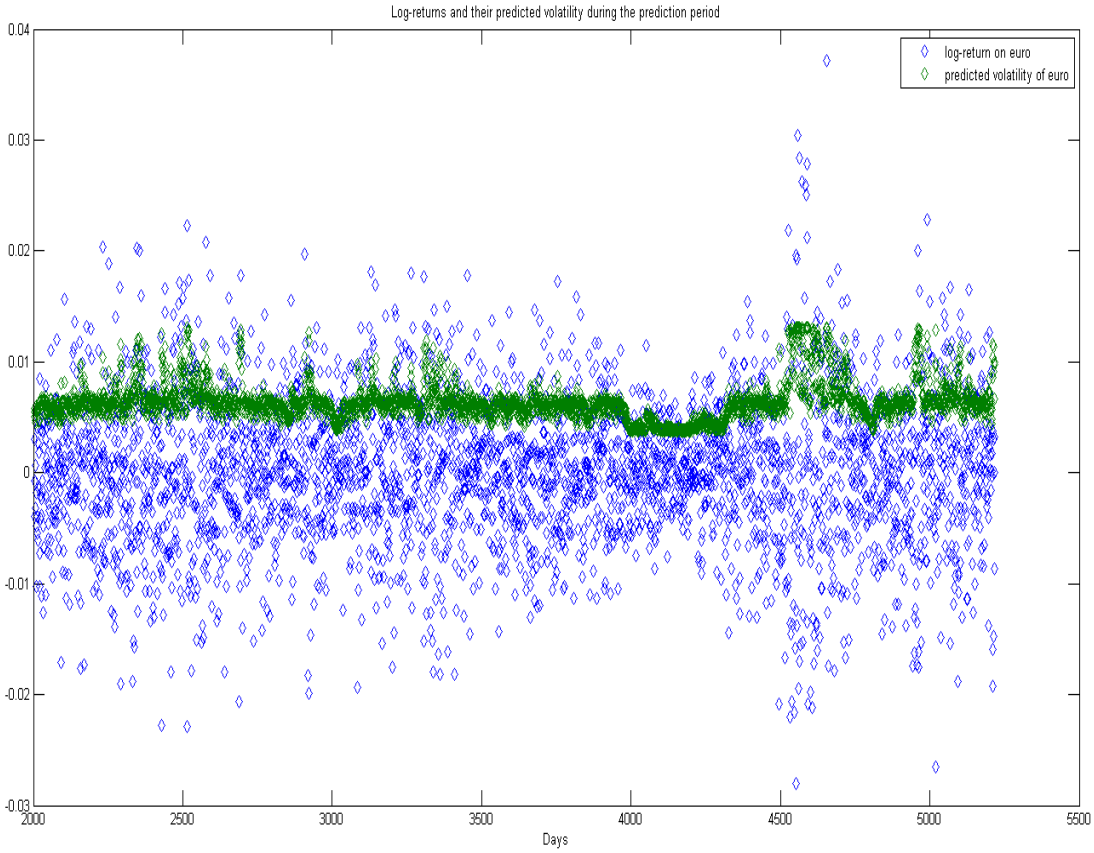
A couple of explanations are possible:

1) The true dynamics of the exchange rate is different between the estimation-validation period and the testing period. No matter how well we estimate a model on the estimation-validation period, it will not capture all the potential moves on the testing period.

2) It may be the case that the whole modeling framework over-predicts the unconditional fatness of tails. Perhaps, we need to think about keeping only two volatility states but let the residuals eps_t have a more complex distribution, not standard normal.

As the final word, we display the expected volatility implied by model 4. It is computed by the forward algorithm and is based on the data from the previous days. This is how we would predict the volatility in reality if we committed to model 4. The expected volatility is juxtaposed with log-returns during the same

(testing) period to illustrate the merits of the model.



Predictive diagnostics is implemented in function `HMM_VolatilityPrediction`().

COMPARISON TO THE ANALYSIS OF LOG-RETURNS ON THE BRITISH POUND

So far we are in the dark regarding the following questions. "How much do the optimal number of states and the optimal number of autoregressive coefficients change if we run the model on a different European currency?" "Are all our insights specific to euro or have we learned anything more general, valid for other economies?" It is not the intention of this study to answer these questions fully. However, we would like to shed some light on the issue. Our findings may serve as a starting point for any in-depth analysis in the future.

We run the already built analytics on the USD/GBP exchange rate during the same time window (May 17, 1991 – May 16, 2011). “GBP” stand for the British pound. We expect the dynamics of pound and euro to be similar, as both must react to events in Europe and worldwide with somewhat similar sensitivities... The complete output of the analysis is placed into appendix C. Here we discuss only the most important results.

Similarly to euro, all coefficients Beta_i are proved to be insignificant. However, the best model is slightly different now, according to the cross-validation errors. Now we have 4 volatility states:

sigma_1 = 1.9159e-005,
sigma_2 = 0.0032,
sigma_3 = 0.0062,
sigma_4 = 0.0145.

The volatility at the first state is tiny. It is an artifact of imperfect likelihood maximization. **The other volatility states**, (sigma_2, sigma_3, sigma_4), **are similar in relative distances to the three volatility states of euro**, which is reassuring. The transition dynamics follows the same rule. **The transition probabilities between states 2 – 4 of pound are similar to those between states 1 – 3 of euro**:

$$A = \begin{matrix} 0.2244 & 0.0000 & 0.7323 & 0.0432 \\ 0.0041 & 0.9529 & 0.0381 & 0.0048 \\ 0.0216 & 0.0309 & 0.9292 & 0.0183 \\ 0.0000 & 0.0000 & 0.1583 & 0.8417 \end{matrix}$$

The results are moderately encouraging. There are similarities between pound and euro. More analysis is necessary for definitive conclusions. **In terms of predictive diagnostics, our models for pound performed much worse than our models for euro** on the testing set. Again, perhaps, the dynamics of the pound has changed recently, or we should modify the models slightly. Here the tiny volatility state (state 1) in the optimal model does not make much weather, as *all 9 models* perform relatively badly.

PredictionInterval_AbsLogError_GBP =

0.4134
0.4216
0.4165
0.3670
0.3745
0.3777
0.3432
0.3886
0.3868

PredictionInterval_VarOfStandResiduals_GBP =

0.6614
0.6560
0.6594
0.6928
0.6877
0.6855
0.7095
0.6780
0.6792

PredictionInterval_Perc_Of_1STD_Events_GBP =

0.2617
0.2575
0.2586
0.2603
0.2589
0.2565
0.2682
0.2534
0.2530

PredictionInterval_Perc_Of_2STD_Events_GBP =

0.0059
0.0059
0.0059
0.0021
0.0021
0.0021
0.0021
0.0052
0.0059

PredictionInterval_Perc_Of_3STD_Events_GBP =

0.0007
0.0007

0.0014
0
0
0
0
0
0

CONCLUSIONS

We make the following conclusions.

- There are three volatility states for the log-returns of euro. The low-volatility state and the medium-volatility state are closer to one another than to the high-volatility state. According to the transition matrix, the world does not spend much time in the high-volatility state. But it may get stuck for quite a while in any of the other two states. The medium-volatility state is most frequent.
- No serial correlation is exhibited by log-returns. The true auto-regressive parameters are zero. This is consistent with financial theory.
- The best model according to the Akaike information criterion (model 8) performs worse on a fresh data set than the best model according to cross-validation (model 4).
- The best model we have identified (model 4) predicts the volatility on a fresh data set with 10% error. This leaves plenty of room for model refinement.
- The estimated models exhibit decent predictive performance when capturing the middle of the range of the future distribution of log-returns. However, all of them are unsatisfactory when predicting the tails of the future distribution. More research has to be done into the optimal choice of the distribution of residuals. Perhaps, possibilities for other types of structural change must be introduced into the models.
- There are many similarities between the dynamics of euro and the British pound. The insights from this study may be cautiously used when modeling other European currencies.

REFERENCES

Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.

Fraser, A. M. (2008). Hidden Markov Models and Dynamical Systems. SIAM.

APPENDIX A: THE MAIN COMMANDS OF THE MATLAB SESSION

```
>> load USD_EUR_rate.txt
>> LogReturn_On_Euro = log(USD_EUR_rate(2:length(USD_EUR_rate))) -
log(USD_EUR_rate(1:(length(USD_EUR_rate)-1)));
>> scatter(1:length(LogReturn_On_Euro),LogReturn_On_Euro)
```

```
>> xlabel('Days')
>> title('Daily log-return on Euro')

>> [BestModel, BestModel_EstimResults, CV_Errors, CV_Errors_For_Blocks, AkaikeInfoCriteria,
PredictionInterval_PredictedVolatility, PredictionInterval_VarOfStandResiduals, PredictionInterval_AbsLogError,
PredictionInterval_Perc_Of_1STD_Events, PredictionInterval_Perc_Of_2STD_Events,
PredictionInterval_Perc_Of_3STD_Events, TrainingPeriod, TrainingData, States, Volatility] =
HMM_Analysis(LogReturn_On_Euro, [0, 1, 2, 0, 1, 2, 0, 1, 2], [2, 2, 2, 3, 3, 3, 4, 4, 4], 2002);

>> scatter(TrainingPeriod,States)
>> xlabel('Days')
>> ylabel('Volatility state')
>> title('The most likely volatility state during the estimation period')
>> plot(TrainingPeriod,Volatility,'d',TrainingPeriod,sqrt(BestModel_EstimResults.VarianceOnTheDay),'d')
>> xlabel('Days')
>> legend('most likely volatility path','expected volatility on a given day')
>> title('Volatility during the estimation period')

>> Model_6 = HMM_Estimation(LogReturn_On_Euro, 2, 3, 3, 2002);

>> load USD_GBP_rate.txt
>> LogReturn_On_GBP = log(USD_GBP_rate(2:length(USD_GBP_rate))) -
log(USD_GBP_rate(1:(length(USD_GBP_rate)-1)));
>> scatter(1:length(LogReturn_On_GBP),LogReturn_On_GBP)
>> [BestModel_GBP, BestModel_EstimResults_GBP, CV_Errors_GBP, CV_Errors_For_Blocks_GBP,
AkaikeInfoCriteria_GBP, PredictionInterval_PredictedVolatility_GBP, PredictionInterval_VarOfStandResiduals_GBP,
PredictionInterval_AbsLogError_GBP, PredictionInterval_Perc_Of_1STD_Events_GBP,
PredictionInterval_Perc_Of_2STD_Events_GBP, PredictionInterval_Perc_Of_3STD_Events_GBP, TrainingPeriod_GBP,
TrainingData_GBP, States_GBP, Volatility_GBP] = HMM_Analysis(LogReturn_On_GBP, [0, 1, 2, 0, 1, 2, 0, 1, 2], [2, 2, 2,
3, 3, 3, 4, 4, 4], 2002);
```

APPENDIX B: SELECTED MATLAB CODE

```
function [BestModel, BestModel_EstimResults, CV_Errors, CV_Errors_For_Blocks, AkaikeInfoCriteria,
PredictionInterval_PredictedVolatility, PredictionInterval_VarOfStandResiduals,
PredictionInterval_AbsLogError, PredictionInterval_Perc_Of_1STD_Events,
PredictionInterval_Perc_Of_2STD_Events, PredictionInterval_Perc_Of_3STD_Events, TrainingPeriod,
TrainingData, States, Volatility] = HMM_Analysis(Series, NumOfLags, NumOfStates, EndOfTrainingData)

%
% The methodology in this analysis follows sections 13.2.1 - 13.2.2, 13.2.6 of
% Bishop's "Pattern Recognition and Machine Learning" as well as sections 2.1 - 2.3
% of Fraser's "Hidden Markov Models and Dynamical Systems".
%

% INITIALIZATION
StartOfTrainingData     = 1 + max(NumOfLags);
if length(NumOfLags) ~= length(NumOfStates)
    fprintf( '\nArrays NumOfLags and NumOfStates must be of the same length, which is the number of
models tried.\n' );
end
NumOfModels             = length(NumOfLags);

% TWO-FOLD CROSS-VALIDATION
fprintf( '\nCROSS-VALIDATION BEGINS.\n');
BlockSize               = floor((EndOfTrainingData-StartOfTrainingData+1) / 2);
CV_Errors_For_Blocks    = zeros(NumOfModels, 2);

for m = 1:NumOfModels
    EstimResults                = HMM_Estimation(Series, NumOfLags(m), NumOfStates(m),
StartOfTrainingData, StartOfTrainingData+BlockSize-1);
    PredictionResults           = HMM_VolatilityPrediction(Series, NumOfLags(m), NumOfStates(m),
EstimResults, StartOfTrainingData+BlockSize, EndOfTrainingData);
    CV_Errors_For_Blocks(m,1)   = PredictionResults.AbsLogError;

    EstimResults                = HMM_Estimation(Series, NumOfLags(m), NumOfStates(m),
StartOfTrainingData+BlockSize, EndOfTrainingData);
    PredictionResults           = HMM_VolatilityPrediction(Series, NumOfLags(m), NumOfStates(m),
EstimResults, StartOfTrainingData, StartOfTrainingData+BlockSize-1);
    CV_Errors_For_Blocks(m,2)   = PredictionResults.AbsLogError;
end
```

```
fprintf( '\nCROSS-VALIDATION COMPLETED.\n');

CV_Errors               = mean(CV_Errors_For_Blocks')';
CV_Errors_Min           = CV_Errors(1) + 1;
for m = 1:NumOfModels
   if CV_Errors(m) < CV_Errors_Min
       CV_Errors_Min    = CV_Errors(m);
       BestModel        = ['Best model: number of lags = ' num2str(NumOfLags(m)) ' and number of
different volatility regimes = ' num2str(NumOfStates(m)) '.'];
       BestModelIndex   = m;
   end
end

% ESTIMATION, AKAIKE CALCULATION & PREDICTIVE PERFORMANCE
AkaikeInfoCriteria                      = zeros(NumOfModels,1);
PredictionInterval_VarOfStandResiduals  = zeros(NumOfModels,1);
PredictionInterval_AbsLogError          = zeros(NumOfModels,1);
PredictionInterval_Perc_Of_1STD_Events  = zeros(NumOfModels,1);
PredictionInterval_Perc_Of_2STD_Events  = zeros(NumOfModels,1);
PredictionInterval_Perc_Of_3STD_Events  = zeros(NumOfModels,1);

for m = 1:NumOfModels
    EstimResults                                = HMM_Estimation(Series, NumOfLags(m),
NumOfStates(m), StartOfTrainingData, EndOfTrainingData);
    AkaikeInfoCriteria(m)                       = EstimResults.AIC;

    PredictionResults                           = HMM_VolatilityPrediction(Series, NumOfLags(m),
NumOfStates(m), EstimResults, EndOfTrainingData+1, length(Series));
    PredictionInterval_PredictedVolatility      = PredictionResults.PredictedVolatility;
    PredictionInterval_VarOfStandResiduals(m)   = PredictionResults.VarianceOfStandResiduals;
    PredictionInterval_AbsLogError(m)           = PredictionResults.AbsLogError;
    PredictionInterval_Perc_Of_1STD_Events(m)   = PredictionResults.Perc_Of_1STD_Events;
    PredictionInterval_Perc_Of_2STD_Events(m)   = PredictionResults.Perc_Of_2STD_Events;
    PredictionInterval_Perc_Of_3STD_Events(m)   = PredictionResults.Perc_Of_3STD_Events;

    if m == BestModelIndex
        BestModel_EstimResults  = EstimResults;
    end
    fprintf( '\nCOMPLETED ESTIMATION & PREDICTION DIAGNOSTICS FOR MODEL %i.\n', m);
end

% IDENTIFYING THE MAXIMUM A POSTERIORI (MAP) STATE SEQUENCE VIA THE VITERBI
% ALGORITHM. THE MAXIMUM A POSTERIORI SEQUENCE IS THE MOST LIKELY SEQUENCE
% GIVEN THE DATA AND THE PARAMETER ESTIMATES.
[TrainingPeriod, TrainingData, States, Volatility]  = HMM_Viterbi(Series, NumOfLags(BestModelIndex),
NumOfStates(BestModelIndex), BestModel_EstimResults, StartOfTrainingData, EndOfTrainingData);
```

```
function PredictionResults = HMM_VolatilityPrediction(RawSeries, NumOfLags, NumOfStates,
EstimResults, StartOfTestingData, EndOfTestingData)

%
% Uses parameter estimates from a different period and predicts volatility
% in the forward looking fashion.
%

AutoRegressCoeff    = EstimResults.AutoRegressCoeff;
VariancesAtStates   = EstimResults.VariancesAtStates;
A                   = EstimResults.A;
APower              = A^100;
StationaryPi        = APower(1,:)';

%
% INITIALIZATION
%
Dim                 = size(RawSeries);
if Dim(1) == 1
    Series  = RawSeries';
else
    Series  = RawSeries;
end
n                   = EndOfTestingData - StartOfTestingData + 1;

X                   = ones(n,1);
for lag = 1:NumOfLags
    X   = [X, Series((StartOfTestingData-lag):(EndOfTestingData-lag))];
end
Y                   = Series(StartOfTestingData:EndOfTestingData);

% FORWARD CALCULATION OF STATE PROBABILITIES
Alpha           = zeros(NumOfStates,n);
Alpha(:,1)      = ( StationaryPi .* 1 ./ sqrt(VariancesAtStates) .* normpdf((Y(1) -
X(1,:)*AutoRegressCoeff) ./ sqrt(VariancesAtStates)) )';
```

```
Alpha(:,1)      = Alpha(:,1) / sum(Alpha(:,1));

for t = 2:n
    Alpha(:,t)      = ( (Alpha(:,t-1)' * A) .* 1 ./ sqrt(VariancesAtStates') .* normpdf((Y(t) -
X(t,:)*AutoRegressCoeff) ./ sqrt(VariancesAtStates')) )';
    Alpha(:,t)      = Alpha(:,t) / sum(Alpha(:,t));
end

% VOLATILITY PREDICTION
PredictedVariance           = Alpha' * VariancesAtStates;
PredictedStandResiduals     = (Y - X*AutoRegressCoeff) ./ sqrt(PredictedVariance);
BurnInPeriod                = min( max(ceil(n/10),150), ceil(n/2) );
VarianceOfStandResiduals    = var(PredictedStandResiduals((BurnInPeriod+1):n));
AbsLogError                 = abs(log( VarianceOfStandResiduals ));
Perc_Of_1STD_Events         = sum( abs(PredictedStandResiduals((BurnInPeriod+1):n)) >= 1 ) / (n -
BurnInPeriod);
Perc_Of_2STD_Events         = sum( abs(PredictedStandResiduals((BurnInPeriod+1):n)) >= 2 ) / (n -
BurnInPeriod);
Perc_Of_3STD_Events         = sum( abs(PredictedStandResiduals((BurnInPeriod+1):n)) >= 3 ) / (n -
BurnInPeriod);

PredictionResults.PredictedVolatility       = sqrt(PredictedVariance);
PredictionResults.VarianceOfStandResiduals  = VarianceOfStandResiduals;
PredictionResults.AbsLogError               = AbsLogError;
PredictionResults.Perc_Of_1STD_Events       = Perc_Of_1STD_Events;
PredictionResults.Perc_Of_2STD_Events       = Perc_Of_2STD_Events;
PredictionResults.Perc_Of_3STD_Events       = Perc_Of_3STD_Events;
PredictionResults.BurnInPeriod              = BurnInPeriod;
```

APPENDIX C: THE COMPLETE MATLAB SESSION

```
>> load USD_EUR_rate.txt
>> LogReturn_On_Euro = log(USD_EUR_rate(2:length(USD_EUR_rate))) -
log(USD_EUR_rate(1:(length(USD_EUR_rate)-1)));
>> scatter(1:length(LogReturn_On_Euro),LogReturn_On_Euro)
>> xlabel('Days')
>> title('Daily log-return on Euro')

>> [BestModel, BestModel_EstimResults, CV_Errors, CV_Errors_For_Blocks, AkaikeInfoCriteria,
PredictionInterval_PredictedVolatility, PredictionInterval_VarOfStandResiduals, PredictionInterval_AbsLogError,
PredictionInterval_Perc_Of_1STD_Events, PredictionInterval_Perc_Of_2STD_Events,
PredictionInterval_Perc_Of_3STD_Events, TrainingPeriod, TrainingData, States, Volatility] =
HMM_Analysis(LogReturn_On_Euro, [0, 1, 2, 0, 1, 2, 0, 1, 2], [2, 2, 2, 3, 3, 3, 4, 4, 4], 2002);

CROSS-VALIDATION BEGINS.

Processed initial guess 1 for the model with 0 lags and 2 states.
Found a better estimate.

Processed initial guess 2 for the model with 0 lags and 2 states.
Found a better estimate.
...

Processed initial guess 1 for the model with 1 lags and 2 states.
Found a better estimate.
...

Processed initial guess 4 for the model with 2 lags and 4 states.

CROSS-VALIDATION COMPLETED.

Processed initial guess 1 for the model with 0 lags and 2 states.
Found a better estimate.

Processed initial guess 2 for the model with 0 lags and 2 states.

Processed initial guess 3 for the model with 0 lags and 2 states.

Processed initial guess 4 for the model with 0 lags and 2 states.

COMPLETED ESTIMATION & PREDICTION DIAGNOSTICS FOR MODEL 1.
```

...

Processed initial guess 1 for the model with 2 lags and 4 states.
Found a better estimate.

Processed initial guess 2 for the model with 2 lags and 4 states.
Found a better estimate.

Processed initial guess 3 for the model with 2 lags and 4 states.

Processed initial guess 4 for the model with 2 lags and 4 states.
Found a better estimate.

COMPLETED ESTIMATION & PREDICTION DIAGNOSTICS FOR MODEL 9.

>> BestModel

BestModel =

Best model: number of lags = 0 and number of different volatility regimes = 3.

>> CV_Errors

CV_Errors =

    0.3038
    0.3037
    0.3023
    0.1905
    0.1922
    0.1914
    0.3213
    0.3354
    0.3211

>> CV_Errors_For_Blocks

CV_Errors_For_Blocks =

    0.5950   0.0126
    0.5953   0.0120
    0.5925   0.0122
    0.3612   0.0199
    0.3611   0.0233
    0.3596   0.0233
    0.6112   0.0313
    0.6141   0.0567
    0.6090   0.0331

>> AkaikeInfoCriteria

AkaikeInfoCriteria =

   -7.4382
   -7.4389
   -7.4387
   -7.4655
   -7.4667
   -7.4665
   -7.4769
   -7.4782
   -7.4776

>> BestModel_EstimResults

BestModel_EstimResults =

              AutoRegressCoeff: -1.1700e-005
    AutoRegressCoeff_StdErrors: 1.3943e-004
           VariancesAtStates: [3x1 double]
            VarianceOnTheDay: [1x2000 double]
                          Pi: [3x1 double]

```
                    A: [3x3 double]
                Alpha: [3x2000 double]
                 Beta: [3x2000 double]
                Gamma: [3x2000 double]
                  Ksi: [3x3x2000 double]
        LogLikelihood: 7.4710e+003
                  AIC: -7.4655
             ExitFlag: 0
                 Iter: 54
          IniGuessCode: 4
ExitFlagsForAllStartingValues: [0 0 0 0]
LogLikelihoodForAllStartingValues: [7.4704e+003 7.4704e+003 7.4705e+003 7.4710e+003]
IterationsForAllStartingValues: [81 73 55 54]

>> BestModel_EstimResults.VariancesAtStates

ans =

  1.0e-003 *

    0.0140
    0.0328
    0.1582

>> BestModel_EstimResults.Pi

ans =

    0.0000
    1.0000
    0.0000

>> BestModel_EstimResults.A

ans =

    0.9864    0.0059    0.0077
    0.0031    0.9735    0.0234
    0.0180    0.1774    0.8046



>> scatter(TrainingPeriod,States)
>> xlabel('Days')
>> ylabel('Volatility state')
>> title('The most likely volatility state during the estimation period')
>> plot(TrainingPeriod,Volatility,'d',TrainingPeriod,sqrt(BestModel_EstimResults.VarianceOnTheDay),'d')
>> xlabel('Days')
>> legend('most likely volatility path','expected volatility on a given day')
>> title('Volatility during the estimation period')



>> PredictionInterval_AbsLogError

PredictionInterval_AbsLogError =

    0.1956
    0.1983
    0.1950
    0.2008
    0.2056
    0.2003
    0.2142
    0.2195
    0.2182

>> PredictionInterval_VarOfStandResiduals

PredictionInterval_VarOfStandResiduals =

    0.8224
```

```
    0.8201
    0.8228
    0.8180
    0.8142
    0.8185
    0.8072
    0.8029
    0.8040

>> PredictionInterval_Perc_Of_1STD_Events

PredictionInterval_Perc_Of_1STD_Events =

    0.3066
    0.3059
    0.3049
    0.3076
    0.3069
    0.3052
    0.2876
    0.2886
    0.2883

>> PredictionInterval_Perc_Of_2STD_Events

PredictionInterval_Perc_Of_2STD_Events =

    0.0093
    0.0090
    0.0093
    0.0028
    0.0028
    0.0028
    0.0121
    0.0090
    0.0104

>> PredictionInterval_Perc_Of_3STD_Events

PredictionInterval_Perc_Of_3STD_Events =

  1.0e-003 *

    0.6913
    0.6913
    0.6913
         0
         0
         0
         0
         0
         0



>>plot(2003:length(LogReturn_On_Euro),LogReturn_On_Euro(2003:length(LogReturn_On_Euro)),'d',2003:length(LogRet
urn_On_Euro),PredictionInterval_PredictedVolatility,'d')
>> xlabel('Days')
>> legend('log-return on euro','predicted volatility of euro')
>> title('Log-returns and their predicted volatility during the prediction period')



>> Model_6 = HMM_Estimation(LogReturn_On_Euro, 2, 3, 3, 2002);

Processed initial guess 1 for the model with 2 lags and 3 states.
Found a better estimate.

Processed initial guess 2 for the model with 2 lags and 3 states.
Found a better estimate.

Processed initial guess 3 for the model with 2 lags and 3 states.
```

```
Processed initial guess 4 for the model with 2 lags and 3 states.
Found a better estimate.
>> Model_6

Model_6 =

                AutoRegressCoeff: [3x1 double]
        AutoRegressCoeff_StdErrors: [3x1 double]
              VariancesAtStates: [3x1 double]
               VarianceOnTheDay: [1x2000 double]
                             Pi: [3x1 double]
                              A: [3x3 double]
                          Alpha: [3x2000 double]
                           Beta: [3x2000 double]
                          Gamma: [3x2000 double]
                            Ksi: [3x3x2000 double]
                 LogLikelihood: 7.4730e+003
                            AIC: -7.4665
                       ExitFlag: 0
                           Iter: 50
                    IniGuessCode: 4
    ExitFlagsForAllStartingValues: [0 0 0 0]
  LogLikelihoodForAllStartingValues: [7.4727e+003 7.4727e+003 7.4727e+003 7.4730e+003]
    IterationsForAllStartingValues: [107 103 66 50]

>> Model_6.AutoRegressCoeff

ans =

   -0.0000
   -0.0262
    0.0193

>> Model_6.AutoRegressCoeff_StdErrors

ans =

    0.0001
    0.0224
    0.0223



>> load USD_GBP_rate.txt
>> LogReturn_On_GBP = log(USD_GBP_rate(2:length(USD_GBP_rate))) -
log(USD_GBP_rate(1:(length(USD_GBP_rate)-1)));
>> scatter(1:length(LogReturn_On_GBP),LogReturn_On_GBP)
>> [BestModel_GBP, BestModel_EstimResults_GBP, CV_Errors_GBP, CV_Errors_For_Blocks_GBP,
AkaikeInfoCriteria_GBP, PredictionInterval_PredictedVolatility_GBP, PredictionInterval_VarOfStandResiduals_GBP,
PredictionInterval_AbsLogError_GBP, PredictionInterval_Perc_Of_1STD_Events_GBP,
PredictionInterval_Perc_Of_2STD_Events_GBP, PredictionInterval_Perc_Of_3STD_Events_GBP, TrainingPeriod_GBP,
TrainingData_GBP, States_GBP, Volatility_GBP] = HMM_Analysis(LogReturn_On_GBP, [0, 1, 2, 0, 1, 2, 0, 1, 2], [2, 2, 2,
3, 3, 3, 4, 4, 4], 2002);

CROSS-VALIDATION BEGINS.

Processed initial guess 1 for the model with 0 lags and 2 states.
Found a better estimate.

...

Processed initial guess 4 for the model with 2 lags and 4 states.

COMPLETED ESTIMATION & PREDICTION DIAGNOSTICS FOR MODEL 9.


>> BestModel_GBP

BestModel_GBP =

Best model: number of lags = 0 and number of different volatility regimes = 4.
```

```
>> CV_Errors_GBP

CV_Errors_GBP =

    0.6659
    0.6669
    0.6653
    0.4097
    0.3890
    0.3874
    0.3331
    0.4385
    0.4401

>> CV_Errors_For_Blocks_GBP

CV_Errors_For_Blocks_GBP =

    0.8548    0.4770
    0.8658    0.4679
    0.8603    0.4703
    0.4792    0.3403
    0.4354    0.3426
    0.4318    0.3430
    0.4085    0.2576
    0.6183    0.2586
    0.6213    0.2589

>> BestModel_EstimResults_GBP

BestModel_EstimResults_GBP =

                AutoRegressCoeff: -1.8215e-005
        AutoRegressCoeff_StdErrors: 1.4387e-004
                VariancesAtStates: [4x1 double]
                 VarianceOnTheDay: [1x2000 double]
                               Pi: [4x1 double]
                                A: [4x4 double]
                            Alpha: [4x2000 double]
                             Beta: [4x2000 double]
                            Gamma: [4x2000 double]
                              Ksi: [4x4x2000 double]
                    LogLikelihood: 7.5686e+003
                              AIC: -7.5591
                         ExitFlag: 0
                             Iter: 80
                      IniGuessCode: 2
     ExitFlagsForAllStartingValues: [0 0 0 0]
   LogLikelihoodForAllStartingValues: [7.5361e+003 7.5686e+003 7.5529e+003 7.5529e+003]
      IterationsForAllStartingValues: [119 80 207 220]

>> BestModel_EstimResults_GBP.VariancesAtStates

ans =

  1.0e-003 *

    0.0000
    0.0102
    0.0388
    0.2093

>> BestModel_EstimResults_GBP.Pi

ans =

         0
    0.0000
    1.0000
    0.0000

>> BestModel_EstimResults_GBP.A
```

ans =

```
    0.2244    0.0000    0.7323    0.0432
    0.0041    0.9529    0.0381    0.0048
    0.0216    0.0309    0.9292    0.0183
    0.0000    0.0000    0.1583    0.8417
```

>> PredictionInterval_AbsLogError_GBP

PredictionInterval_AbsLogError_GBP =

```
    0.4134
    0.4216
    0.4165
    0.3670
    0.3745
    0.3777
    0.3432
    0.3886
    0.3868
```

>> PredictionInterval_VarOfStandResiduals_GBP

PredictionInterval_VarOfStandResiduals_GBP =

```
    0.6614
    0.6560
    0.6594
    0.6928
    0.6877
    0.6855
    0.7095
    0.6780
    0.6792
```

>> PredictionInterval_Perc_Of_1STD_Events_GBP

PredictionInterval_Perc_Of_1STD_Events_GBP =

```
    0.2617
    0.2575
    0.2586
    0.2603
    0.2589
    0.2565
    0.2682
    0.2534
    0.2530
```

>> PredictionInterval_Perc_Of_2STD_Events_GBP

PredictionInterval_Perc_Of_2STD_Events_GBP =

```
    0.0059
    0.0059
    0.0059
    0.0021
    0.0021
    0.0021
    0.0021
    0.0052
    0.0059
```

>> PredictionInterval_Perc_Of_3STD_Events_GBP

PredictionInterval_Perc_Of_3STD_Events_GBP =

```
    0.0007
    0.0007
    0.0014
         0
         0
```

0
0
0
0
