An Online Vector Error Correction Model for Exchange Rates Forecasting

Paola Arce[1], Jonathan Antognini[1], Werner Kristjanpoller[2] and Luis Salinas[1,3]
[1]*Departamento de Informática, Universidad Técnica Federico Santa María, Valparaíso, Chile*
[2]*Departamento de Industrias, Universidad Técnica Federico Santa María, Valparaíso, Chile*
[3]*CCTVal, Universidad Técnica Federico Santa María, Valparaíso, Chile*

Keywords: Vector Error Correction Model, Online Learning, Financial Forecasting, Foreign Exchange Market.

Abstract: Financial time series are known for their non-stationary behaviour. However, sometimes they exhibit some stationary linear combinations. When this happens, it is said that those time series are cointegrated.The Vector Error Correction Model (VECM) is an econometric model which characterizes the joint dynamic behaviour of a set of cointegrated variables in terms of forces pulling towards equilibrium. In this study, we propose an Online VEC model (OVECM) which optimizes how model parameters are obtained using a sliding window of the most recent data. Our proposal also takes advantage of the long-run relationship between the time series in order to obtain improved execution times. Our proposed method is tested using four foreign exchange rates with a frequency of 1-minute, all related to the USD currency base. OVECM is compared with VECM and ARIMA models in terms of forecasting accuracy and execution times. We show that OVECM outperforms ARIMA forecasting and enables execution time to be reduced considerably while maintaining good accuracy levels compared with VECM.

1 INTRODUCTION

In finance, it is common to find variables with long-run equilibrium relationships. This is called cointegration and it reflects the idea of that some set of variables cannot wander too far from each other. Cointegration means that one or more linear combinations of these variables are stationary even though individually they are not (Engle and Granger, 1987). Furthermore, the number of cointegration vectors reflects how many of these linear combinations exist. Some models, such as the Vector Error Correction (VECM), take advantage of this property and describe the joint behaviour of several cointegrated variables.

VECM introduces this long-run relationship among a set of cointegrated variables as an error correction term. VECM is a special case of the vector autorregresive model (VAR) model. VAR model expresses future values as a linear combination of variables past values. However, VAR model cannot be used with non-stationary variables. VECM is a linear model but in terms of variable differences. If cointegration exists, variable differences are stationary and they introduce an error correction term which adjusts coefficients to bring the variables back to equilibrium. In finance, many economic time series are revealed to be stationary when they are differentiated and cointegration restrictions often improves forecasting (Duy and Thoma, 1998). Therefore, VECM has been widely adopted.

In finance, pair trading is a very common example of cointegration application (Herlemont, 2003) but cointegration can also be extended to a larger set of variables (Mukherjee and Naka, 1995),(Engle and Patton, 2004).

Both VECM and VAR model parameters are obtained using ordinary least squares (OLS) method. Since OLS involves many calculations, the parameter estimation method is computationally expensive when the number of past values and observations increases. Moreover, obtaining cointegration vectors is also an expensive routine.

Recently, online learning algorithms have been proposed to solve problems with large data sets because of their simplicity and their ability to update the model when new data is available. The study presented by (Arce and Salinas, 2012) applied this idea using ridge regression.

There are several popular online methods such as perceptron (Rosenblatt, 1958), passive-

Arce P., Antognini J., Kristjanpoller W. and Salinas L..
An Online Vector Error Correction Model for Exchange Rates Forecasting.
DOI: 10.5220/0005205901930200
In *Proceedings of the International Conference on Pattern Recognition Applications and Methods* (ICPRAM-2015), pages 193-200
ISBN: 978-989-758-077-2

aggressive (Crammer et al., 2006), stochastic gradient descent (Zhang, 2004), aggregating algorithm (Vovk, 2001) and the second order perceptron (Cesa-Bianchi et al., 2005). In (Cesa-Bianchi and Lugosi, 2006), an in-deph analysis of online learning is provided.

In this paper, we propose an online formulation of the VECM called Online VECM (OVECM). OVECM is a lighter version of VECM which considers only a sliding window of the most recent data and introduces matrix optimizations in order to reduce the number of operations and therefore execution times. OVECM also takes into account the fact that cointegration vector space doesn't experience large changes with small changes in the input data.

OVECM is later compared against VECM and ARIMA models using four currency rates from the foreign exchange market with 1-minute frequency. VECM and ARIMA models were used in an iterative way in order to allow fair comparison. Execution times and forecast performance measures MAPE, MAE and RMSE were used to compare all methods.

Model effectiveness is focused on out-of-sample forecast rather than in-sample fitting. This criteria allows the OVECM prediction capability to be expressed rather than just explaining data history.

The next sections are organized as follows: section 2 presents the VAR and VECM, the OVECM algorithm proposed is presented in section 3. Section 4 gives a description of the data used and the tests carried on to show accuracy and time comparison of our proposal against the traditional VECM and section 5 includes conclusions and a proposal for future study.

2 BACKGROUND

2.1 Integration and Cointegration

A time series $\mathbf{y}$ is said to be integrated of order d if after differentiating the variable d times, we get an I(0) process, more precisely:

$$(1-L)^d \mathbf{y} \sim \mathrm{I}(0),$$

where I(0) is a stationary time series and L is the lag operator:

$$(1-L)\mathbf{y} = \Delta\mathbf{y} = \mathbf{y}_t - \mathbf{y}_{t-1} \quad \forall t$$

Let $\mathbf{y}_t = \{\mathbf{y}^1, \ldots, \mathbf{y}^l\}$ be a set of l stationary time series I(1) which are said to be cointegrated if a vector $\beta = [\beta(1), \ldots, \beta(l)]^\top \in \mathbb{R}^l$ exists such that the time series,

$$\mathbf{Z}_t := \beta^\top \mathbf{y}_t = \beta(1)\mathbf{y}^1 + \cdots + \beta(l)\mathbf{y}^l \sim \mathrm{I}(0). \quad (1)$$

In other words, a set of I(1) variables is said to be cointegrated if a linear combination of them exists which is I(0).

2.2 Vector Autorregresive Models

VECM is a special case of VAR model and both describe the joint behaviour of a set of variables.

VAR(p) model is a general framework to describe the behaviour of a set of l endogenous variables as a linear combination of their last p values. These l variables at time t are represented by the vector $\mathbf{y}_t$ as follows:

$$\mathbf{y}_t = \begin{bmatrix} y_{1,t} & y_{2,t} & \cdots & y_{l,t} \end{bmatrix}^\top,$$

where $y_{j,t}$ corresponds to the time series j evaluated at time t.

The VAR(p) model describes the behaviour of a dependent variable in terms of its own lagged values and the lags of the others variables in the system. The model with p lags is formulated as the following:

$$\mathbf{y}_t = \phi_1 \mathbf{y}_{t-1} + \cdots + \phi_p \mathbf{y}_{t-p} + \mathbf{c} + \varepsilon_t, \quad (2)$$

where $t = p+1, \ldots N$, $\phi_1, \ldots, \phi_p$ are $l \times l$ matrices of real coefficients, $\varepsilon_{p+1}, \ldots, \varepsilon_N$ are error terms, $\mathbf{c}$ is a constant vector and N is the total number of samples.

The VAR matrix form of equation (2) is:

$$\mathbf{B} = \mathbf{A}\mathbf{X} + \mathbf{E}, \quad (3)$$

where:

$$\mathbf{A} = \begin{bmatrix} \mathbf{y}_p & \cdots & \mathbf{y}_{N-1} \\ \mathbf{y}_{p-1} & \cdots & \mathbf{y}_{N-2} \\ \vdots & \ddots & \vdots \\ \mathbf{y}_1 & \cdots & \mathbf{y}_{N-p} \\ 1 & \cdots & 1 \end{bmatrix}^\top,$$

$$\mathbf{B} = \begin{bmatrix} \mathbf{y}_{p+1} & \cdots & \mathbf{y}_N \end{bmatrix}^\top,$$

$$\mathbf{X} = \begin{bmatrix} \phi_1 & \cdots & \phi_p & \mathbf{c} \end{bmatrix}^\top,$$

$$\mathbf{E} = \begin{bmatrix} \varepsilon_{p+1} & \cdots & \varepsilon_N \end{bmatrix}^\top.$$

Equation (3) can be solved using ordinary least squares estimation.

2.3 VECM

VECM is a special form of a VAR model for I(1) variables that are also cointegrated (Banerjee, 1993). It is obtained by replacing the form $\Delta \mathbf{y}_t = \mathbf{y}_t - \mathbf{y}_{t-1}$ in equation (2). VECM is expressed in terms of differences, has an error correction term and the following form:

$$\Delta \mathbf{y}_t = \underbrace{\Omega \mathbf{y}_{t-1}}_{\text{Error correction term}} + \sum_{i=1}^{p-1} \phi_i^* \Delta \mathbf{y}_{t-i} + \mathbf{c} + \varepsilon_t \quad , \quad (4)$$

where coefficients matrices Ω and ϕ_i^* are function of matrices ϕ_i (shown in equation (2)) as follows:

$$\begin{aligned} \phi_i^* &:= -\sum_{j=i+1}^{p} \phi_j \\ \Omega &:= -(\mathbb{I} - \phi_1 - \cdots - \phi_p) \end{aligned}$$

The matrix Ω has the following properties (Johansen, 1995):

- If $\Omega = 0$ there is no cointegration
- If $rank(\Omega) = l$ i.e full rank, then the time series are not I(1) but stationary
- If $rank(\Omega) = r, \quad 0 < r < l$ then, there is cointegration and the matrix Ω can be expressed as $\Omega = \alpha\beta^\top$, where α and β are $(l \times r)$ matrices and $rank(\alpha) = rank(\beta) = r$.

  The columns of β contains the cointegration vectors and the rows of α correspond with the adjusted vectors. β is obtained by Johansen procedure (Johansen, 1988) whereas α has to be determined as a variable in the VECM.

  It is worth noticing that the factorization of the matrix Ω is not unique since for any $r \times r$ nonsingular matrix H we have:

$$\begin{aligned} \alpha\beta^\top &= \alpha \mathbf{H}\mathbf{H}^{-1}\beta^\top \\ &= (\alpha\mathbf{H})(\beta(\mathbf{H}^{-1})^\top)^\top \\ &= \alpha^*(\beta^*)^\top \end{aligned}$$

  with $\alpha^* = \alpha\mathbf{H}$ and $\beta^* = \beta(\mathbf{H}^{-1})^\top$.

If cointegration exists, then equation (4) can be written as follows:

$$\Delta \mathbf{y}_t = \alpha\beta^\top \mathbf{y}_{t-1} + \sum_{i=1}^{p-1} \phi_i^* \Delta \mathbf{y}_{t-i} + \mathbf{c} + \varepsilon_t \quad ,$$

which is a VAR model but for time series differences.

VECM has the same form shown in equation (3) but with different matrices:

$$\mathbf{A} = \begin{bmatrix} \beta^\top \mathbf{y}_p & \cdots & \beta^\top \mathbf{y}_{N-1} \\ \Delta \mathbf{y}_p & \cdots & \Delta \mathbf{y}_{N-1} \\ \vdots & \ddots & \vdots \\ \Delta \mathbf{y}_2 & \cdots & \Delta \mathbf{y}_{N-p+1} \\ 1 & \cdots & 1 \end{bmatrix}^\top , \quad (5)$$

$$\mathbf{B} = \begin{bmatrix} \Delta \mathbf{y}_{p+1} & \cdots & \Delta \mathbf{y}_N \end{bmatrix}^\top , \quad (6)$$

$$\mathbf{X} = \begin{bmatrix} \alpha & \phi_1^* & \cdots & \phi_{p-1}^* & \mathbf{c} \end{bmatrix}^\top , \quad (7)$$

$$\mathbf{E} = \begin{bmatrix} \varepsilon_{p+1} & \cdots & \varepsilon_N \end{bmatrix}^\top \quad (8)$$

VAR and VECM parameters shown in equation (3) can be solved using standard regression techniques, such as ordinary least squares (OLS).

2.4 Ordinary Least Squares Method

When $\mathbf{A}$ is singular, solution to equation (3) is given by the ordinary least squares (OLS) method. OLS consists of minimizing the sum of squared errors or equivalently minimizing the following expression:

$$\min_{\mathbf{X}} \quad \|\mathbf{A}\mathbf{X} - \mathbf{B}\|_2^2$$

for which the solution $\hat{\mathbf{X}}$ is well-known:

$$\hat{\mathbf{X}} = \mathbf{A}^+ \mathbf{B}$$

where $\mathbf{A}^+$ is the Moore-Penrose pseudo-inverse which can be written as follows:

$$\mathbf{A}^+ = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top . \quad (9)$$

However, when $\mathbf{A}$ is not full rank, i.e $rank(\mathbf{A}) = k < n \leq m$, $\mathbf{A}^\top \mathbf{A}$ is always singular and equation (9) cannot be used. More generally, the pseudo-inverse is best computed using the compact singular value decomposition (SVD) of $\mathbf{A}$:

$$\underset{m \times n}{\mathbf{A}} = \underset{m \times k}{\mathbf{U_1}} \; \underset{k \times k}{\Sigma_1} \; \underset{k \times n}{\mathbf{V}_1^\top} ,$$

as follows

$$\mathbf{A}^+ = \mathbf{V}_1 \Sigma_1^{-1} \mathbf{U}_1^\top .$$

3 METHODOLOGY

3.1 Online VECM

Since VECM parameter estimation is computationally expensive, we propose an online version of VECM (OVECM). OVECM considers only a sliding window of the most recent data. Moreover, since cointegration vectors represent long-run relationships which vary little in time, OVECM determines firstly if they require calculation.

OVECM also implements matrix optimizations in order to reduce execution time, such as updating matrices with new data, removing old data and introducing new cointegration vectors.

Algorithm 1 shows our OVECM proposal which considers the following:

- The function `getJohansen` returns cointegration vectors given by the Johansen method considering the trace statistic test at 95% significance level.
- The function `vecMatrix` returns the matrices (5) and (6) that allows VECM to be solved.
- The function `vecMatrixOnline` returns the matrices (5) and (6) aggregating new data and removing the old one, avoiding calculation of the matrix **A** from scratch.
- Out-of-sample outputs are saved in the variables $\mathbf{Y}_{\text{true}}$ and $\mathbf{Y}_{\text{pred}}$.
- The model is solved using OLS.
- In-sample outputs are saved in the variables $\Delta\mathbf{y}_{\text{true}}$ and $\Delta\mathbf{y}_{\text{pred}}$.
- The function `mape` gets the in-sample MAPE for the l time series.
- Cointegration vectors are obtained at the beginning and when they are required to be updated. This updating is decided based on the in-sample MAPE of the last n inputs. The average of all MAPEs are stored in the variable e. If the average of MAPEs (`mean`(e)) is above a certain error given by the mean_error threshold, cointegration vectors are updated.
- If new cointegration vectors are required, the function `vecMatrixUpdate` only updates the corresponding columns of matrix **A** affected by those vectors (see equation 5).

Our proposal was compared against VECM and ARIMA. Both algorithms were adapted to an online context in order to get a reasonable comparison with our proposal (see algorithms 2 and 3). VECM and ARIMA are called with a sliding window of the most recent data, whereby the models are updated at every time step.

Since we know out time series are I(1) SLARIMA is called with $d = 1$. ARIMA is executed for every time series.

Both OVECM and SLVECM time complexity is dominated by Johansen method which is $O(n^3)$. Thus, both algorithms order is $O(Cn^3)$ where C is the number of iterations.

Algorithm 1: OVECM: Online VECM.

Input:
- **y**: matrix with N input vectors and l time series
- p: number of past values
- L: sliding window size ($L < N$)
- mean_error: MAPE threshold
- n: interpolation points to obtain MAPE

Output:
- $\{\mathbf{y}_{\text{pred}}[L+1],\ldots,\mathbf{y}_{\text{pred}}[N]\}$: model predictions

1: **for** $i = 0$ to $N-L$ **do**
2: $\quad\mathbf{y}_i \leftarrow \mathbf{y}[i : i+L]$
3: $\quad$**if** i = 0 **then**
4: $\quad\quad v \leftarrow \texttt{getJohansen}(\mathbf{y}_i, p)$
5: $\quad\quad [\mathbf{A}\quad \mathbf{B}] \leftarrow \texttt{vecMatrix}(\mathbf{y}_i, p, v)$
6: $\quad$**else**
7: $\quad\quad [\mathbf{A}\quad \mathbf{B}] \leftarrow \texttt{vecMatrixOnline}(\mathbf{y}_i, p, v, \mathbf{A}, \mathbf{B})$
8: $\quad\quad \Delta\mathbf{Y}_{\text{pred}}[i] \leftarrow \mathbf{A}[-1,:] \times \mathbf{X}$
9: $\quad$**end if**
10: $\quad\mathbf{X} \leftarrow \texttt{OLS}(\mathbf{A}, \mathbf{B})$
11: $\quad e \leftarrow \texttt{mape}(\mathbf{B}[-n,:], \mathbf{A}[-n,:] \times \mathbf{X})$
12: $\quad$**if** $\texttt{mean}(e) >$ mean_error **then**
13: $\quad\quad v \leftarrow \texttt{getJohansen}(\mathbf{y}_i, p)$
14: $\quad\quad \mathbf{A} \leftarrow \texttt{vecMatrixUpdate}(\mathbf{y}_i, p, v, \mathbf{A})$
15: $\quad\quad \mathbf{X} \leftarrow \texttt{OLS}(\mathbf{A}, \mathbf{B})$
16: $\quad$**end if**
17: **end for**
18: $\mathbf{Y}_{\text{true}} \leftarrow \mathbf{Y}[L+1 : N]$
19: $\mathbf{Y}_{\text{pred}} \leftarrow \mathbf{Y}[L : N-1] + \Delta\mathbf{Y}_{\text{pred}}$

Algorithm 2: SLVECM: Sliding window VECM.

Input:
- **y**: matrix with N input vectors and l time series
- p: number of past values
- L: sliding window size ($L < N$)

Output:
- $\{\mathbf{y}_{\text{pred}}[L+1],\ldots,\mathbf{y}_{\text{pred}}[N]\}$: model predictions

1: **for** $i = 0$ to $N-L$ **do**
2: $\quad\mathbf{y}_i \leftarrow \mathbf{y}[i : i+L+1]$
3: $\quad model = VECM(\mathbf{y}_i, p)$
4: $\quad\mathbf{Y}_{\text{pred}}[i] = model.predict(\mathbf{y}[i+L])$
5: **end for**
6: $\mathbf{Y}_{\text{true}} = \mathbf{y}[i+L+1 : N]$

Algorithm 3: SLARIMA: Sliding window ARIMA.

Input:
$\mathbf{y}$: matrix with N input vectors and l time series
p: autoregressive order
d: integrated order
q: moving average order
L: sliding window size ($L < N$)

Output:
$\{\mathbf{y}_{\text{pred}}[L+1], \ldots, \mathbf{y}_{\text{pred}}[N]\}$: model predictions

for $i = 0$ to $N-L$ **do**
  for $j = 0$ to $l-1$ **do**
    $\mathbf{y}_i \leftarrow \mathbf{y}[i : i+L+1, j]$
    $model = ARIMA(\mathbf{y}_i, (p,d,q))$
    $\mathbf{Y}_{\text{pred}}[i,j] = model.predict(\mathbf{y}[i+L, j])$
  end for
end for
$\mathbf{Y}_{\text{true}} = \mathbf{y}[i+L+1 : N, :]$

3.2 Evaluation Methods

Forecast performance is evaluated using different methods. We have chosen three measures usually used:

MAPE. Mean Average Percent Error which presents forecast errors as a percentage.

$$\text{MAPE} = \frac{1}{N}\sum_{t=1}^{N}\frac{|\mathbf{y}_t - \hat{\mathbf{y}}_t|}{|\mathbf{y}_t|} \times 100 \qquad (10)$$

MAE. Mean Average Error which measures the distance between forecasts to the true value.

$$\text{MAE} = \frac{1}{N}\sum_{t=1}^{N}|\mathbf{y}_t - \hat{\mathbf{y}}_t| \qquad (11)$$

RMSE. Root Mean Square Error also measures the distance between forecasts to the true values but, unlike MAE, large deviations from the true value have a large impact on RMSE due to squaring forecast error.

$$\text{RMSE} = \sqrt{\frac{\sum_{t=1}^{N}(\mathbf{y}_t - \hat{\mathbf{y}}_t)^2}{N}} \qquad (12)$$

3.3 Model Selection

Akaike Information Criterion (AIC) is often used in model selection where AIC with smaller values are preferred since they represent a trade-off between bias and variance. AIC is obtained as follows:

$$AIC = \underbrace{-\frac{2l}{N}}_{\text{bias}} + \underbrace{\frac{2k}{N}}_{\text{variance}} \qquad (13)$$

where

l is the loglikelihood function

k number of estimated parameters

N number of observations

Loglikelihood function is obtained from the Residual Sum of Squares (RSS):

$$l = -\frac{N}{2}\left(1 + ln(2\pi) + ln\left(\frac{RSS}{N}\right)\right) \qquad (14)$$

4 RESULTS

4.1 Data

Tests of SLVECM, SLARIMA and our proposal OVECM were carried out using foreign four exchange data rates: EURUSD, GBPUSD, USDCHF and USDJPY. This data was collected from the free database Dukascopy which gives access to the Swiss Foreign Exchange Marketplace (Dukascopy, 2014).

The reciprocal of the last two rates (CHFUSD, JPYUSD) were used in order to obtain the same base currency for all rates. The tests were done using 1-minute frequency from ask prices which corresponded to 1.440 data points per day from the 11th to the 15th of August 2014.

4.2 Unit Root Tests

Before running the tests, we firstly checked if they were I(1) time series using the Augmented Dickey Fuller (ADF) test.

Table 1: Unit roots tests.

	Statistic	Critical value	Result
EURUSD	-0.64	-1.94	True
ΔEURUSD	-70.45	-1.94	False
GBPUSD	-0.63	-1.94	True
ΔGBPUSD	-54.53	-1.94	False
CHFUSD	-0.88	-1.94	True
ΔCHFUSD	-98.98	-1.94	False
JPYUSD	-0.65	-1.94	True
ΔJPYUSD	-85.78	-1.94	False

Table 1 shows that all currency rates cannot reject the unit root test but they rejected it with their first differences. This means that all of them are I(1) time series and we are allowed to use VECM and therefore OVECM.

4.3 Parameter Selection

In order to set OVECM parameters: windows size L and lag order p, we propose to use several window sizes: $L = 100, 400, 700, 1000$. For every window size L we chose lag order with minimum AIC.

ARIMA parameters were also obtained using AIC. Parameters optimization is presented in table 2:

Table 2: Parameters optimization. VECM order and ARIMA parameters were selected using AIC.

Windows size L L	VECM order (p)	ARIMA order (p,d,q)
100	2	p=2,d=1,q=1
400	5	p=1,d=1,q=1
700	3	p=2,d=1,q=1
1000	3	p=2,d=1,q=1

In OVECM we also define a mean_error variable, which was defined based on the in-sample MAPEs. Figure 1 shows how MAPE moves and how mean_error variable help us to decide whether new cointegration vectors are needed.

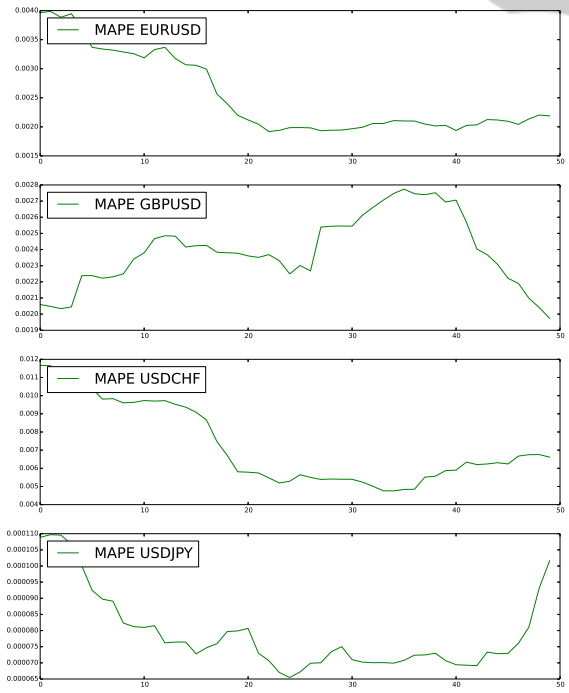

Figure 1: In-sample MAPEs example for 50 minutes. The average of them is considered to obtain new cointegration vectors.

4.4 Execution Times

We ran OVECM and SLVECM 400 iterations. SLARIMA execution time is excluded because its is not comparable with OVECM and SLVECM. SLARIMA was created based on statsmodels library routine ARIMA.

The execution times are shown in the table 3.

Table 3: Execution times.

	L	order	e	Time[s]
OVECM	100	p=2	0	2.492
OVECM	100	p=2	0.0026	1.606
SLVECM	100	p=2	–	2.100
OVECM	400	p=5	0	3.513
OVECM	400	p=5	0.0041	2.569
SLVECM	400	p=5	–	3.222
OVECM	700	p=3	0	3.296
OVECM	700	p=3	0.0032	2.856
SLVECM	700	p=3	–	3.581
OVECM	1000	p=3	0	4.387
OVECM	1000	p=3	0.0022	2.408
SLVECM	1000	p=3	–	3.609

It is clear that execution time depends directly on L and p since they determine the size of matrix $\mathbf{A}$ and therefore affect the OLS function execution time. It is worthy of note that execution time also depends on mean_error because it determines how many times OVECM will recalculate cointegration vectors which is an expensive routine.

Figure 1 shows an example of the in-sample MAPE for 50 iterations. When the average of the in-sample MAPEs is above mean_error new cointegration vectors are obtained. In consequence, OVECM performance increases when mean_error increases. However, this could affect accuracy, but table 4 shows that using an appropriate mean_error doesn't affect accuracy considerable.

4.5 Performance Accuracy

Table 4 shows in-sample and out-of-sample performance measures: MAPE, MAE and RMSE for OVECM, SLVECM and SLARIMA. Test were done using the parameters defined in table 2. We can see that OVECM has very similar performance than SLVECM and this support the theory that cointegration vectors vary little in time. Moreover, OVECM also out performed SLARIMA based on these three performance measures.

We can also notice that in-sample performance in OVECM and SLVECM is related with the out-of-sample performance. This differs with SLARIMA

Table 4: Model measures.

Model				In-sample			Out-of-sample		
Method	L	Parameters	e	MAPE	MAE	RMSE	MAPE	MAE	RMSE
OVECM	100	P=2	0.0026	0.00263	0.00085	0.00114	0.00309	0.00094	0.00131
OVECM	400	P=5	0.0041	0.00378	0.00095	0.00127	0.00419	0.00103	0.00143
OVECM	700	P=3	0.0032	0.00323	0.00099	0.00130	0.00322	0.00097	0.00132
OVECM	1000	P=3	0.0022	**0.00175**	**0.00062**	**0.00087**	**0.00172**	**0.00061**	**0.00090**
SLVECM	100	P=2	-	0.00262	0.00085	0.00113	0.00310	0.00095	0.00132
SLVECM	400	P=5	-	0.00375	0.00095	0.00126	0.00419	0.00103	0.00143
SLVECM	700	P=3	-	0.00324	0.00099	0.00130	0.00322	0.00098	0.00132
SLVECM	1000	P=3	-	**0.00174**	**0.00061**	**0.00087**	**0.00172**	**0.00061**	**0.00090**
SLARIMA	100	p=2, d=1, q=1	-	0.00285	0.00110	0.00308	0.00312	0.00098	0.00144
SLARIMA	400	p=1, d=1, q=1	-	0.00377	0.00101	0.00128	0.00418	0.00106	0.00145
SLARIMA	700	p=2, d=1, q=1	-	0.00329	0.00102	0.00136	0.00324	0.00097	0.00133
SLARIMA	1000	p=2, d=1, q=1	-	**0.00281**	**0.00074**	**0.00105**	**0.00177**	**0.00063**	**0.00092**

which models with good in-sample performance are not necessarily good out-of-sample models. Moreover OVECM outperformed SLARIMA using the same window size.

Figure 2 shows the out-of-sample forecasts made by our proposal OVECM with the best parameters found based on table 4 which follows the time series very well.

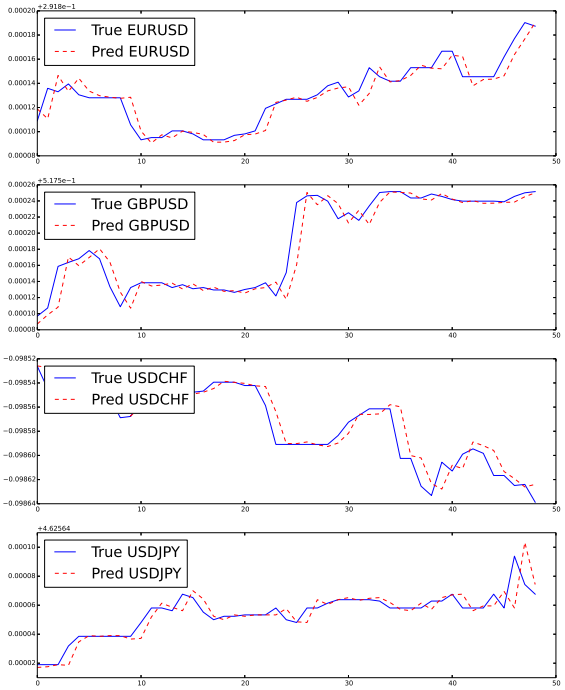

Figure 2: OVECM forecasting accuracy example for 50 minutes using $L = 1000$ and $p = 3$.

5 CONCLUSIONS

A new online vector error correction method was presented. We have shown that our proposed OVECM considerably reduces execution times without compromising solution accuracy. OVECM was compared with VECM and ARIMA with the same sliding window sizes and OVECM outperformed both in terms of execution time. Traditional VECM slightly outperformed our proposal but the OVECM execution time is lower. This reduction of execution time is mainly because OVECM avoids the cointegration vector calculation using the Johansen method. The condition for getting new vectors is given by the mean_error variable which controls how many times the Johansen method is called. Additionally, OVECM introduces matrix optimization in order to get the new model in an iterative way. We could see that our algorithm took much less than a minute at every step. This means that it could also be used with higher frequency data and would still provide responses before new data arrives.

For future study, it would be interesting to improve the out-of-sample forecast by considering more explicative variables, to increase window sizes or trying new conditions to obtain new cointegration vectors.

Since OVECM is an online algorithm which optimizes processing time, it could be used by investors as an input for strategy robots. Moreover, some technical analysis methods could be based on its output.

REFERENCES

Arce, P. and Salinas, L. (2012). Online ridge regression method using sliding windows. *2011 30th International Conference of the Chilean Computer Science Society*, 0:87–90.

Banerjee, A. (1993). *Co-integration, Error Correction, and the Econometric Analysis of Non-stationary Data*. Advanced texts in econometrics. Oxford University Press.

Cesa-Bianchi, N., Conconi, A., and Gentile, C. (2005). A second-order perceptron algorithm. *SIAM J. Comput.*, 34(3):640–668.

Cesa-Bianchi, N. and Lugosi, G. (2006). *Prediction, Learning, and Games*. Cambridge University Press, New York, NY, USA.

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., and Singer, Y. (2006). Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.

Dukascopy (2014). Dukascopy historical data feed.

Duy, T. A. and Thoma, M. A. (1998). Modeling and forecasting cointegrated variables: some practical experience. *Journal of Economics and Business*, 50(3):291–307.

Engle, R. F. and Granger, C. W. J. (1987). Co-Integration and Error Correction: Representation, Estimation, and Testing. *Econometrica*, 55(2):251–276.

Engle, R. F. and Patton, A. J. (2004). Impacts of trades in an error-correction model of quote prices. *Journal of Financial Markets*, 7(1):1–25.

Herlemont, D. (2003). Pairs trading, convergence trading, cointegration. *YATS Finances & Techonologies*, 5.

Johansen, S. (1988). Statistical analysis of cointegration vectors. *Journal of Economic Dynamics and Control*, 12(2-3):231–254.

Johansen, S. (1995). *Likelihood-Based Inference in Cointegrated Vector Autoregressive Models*. Oxford University Press.

Mukherjee, T. K. and Naka, A. (1995). Dynamic relations between macroeconomic variables and the japanese stock market: an application of a vector error correction model. *Journal of Financial Research*, 18(2):223–37.

Rosenblatt, F. (1958). The Perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408.

Vovk, V. (2001). Competitive on-line statistics. *International Statistical Review*, 69:2001.

Zhang, T. (2004). Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *ICML 2004: Proceedings of the twenty-first international conference on Machine Learning. OMNIPRESS*, pages 919–926.